TESIS – SS142501

BOOSTING SUPPORT VECTOR MACHINE PADA DATA *MICROARRAY* YANG *IMBALANCE*

RISKY FRASETIO WAHYU PRATAMA
NRP. 06211650010002

DOSEN PEMBIMBING
Santi Wulan Purnami, M.Si., Ph.D.
Dr. Santi Puteri Rahayu, M.Si.

PROGRAM MAGISTER
JURUSAN STATISTIKA
FAKULTAS MATEMATIKA, KOMPUTASI DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

THESIS – SS14 2501

BOOSTING SUPPORT VECTOR MACHINE FOR IMBALANCED MICROARRAY DATA

RISKY FRASETIO WAHYU PRATAMA
NRP. 06211650010002

SUPERVISORS
Santi Wulan Purnami, M.Si., Ph.D.
Dr. Santi Puteri Rahayu, M.Si.

MAGISTER PROGRAMME
DEPARTMENT OF STATISTICS
FACULTY OF MATHEMATICS, COMPUTING AND DATA SCIENCES
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

BOOSTING SUPPORT VECTOR MACHINE PADA DATA MICROARRAY YANG IMBALANCE

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Master Sains (M.Si)
di
Institut Teknologi Sepuluh Nopember

oleh :
RISKY FRASETIO WAHYU PRATAMA
NRP. 06211650010002

Tanggal Ujian : 09 Juli 2018
Periode Wisuda : September 2018

Disetujui oleh:

1. **Santi Wulan Purnami, M.Si., Ph.D.** **(Pembimbing I)**
   NIP: 19720923 199803 2 001

2. **Dr. Santi Puteri Rahayu, M.Si.** **(Pembimbing II)**
   NIP: 19750115 199903 2 003

3. **Dr. Bambang Widjanarko Otok, M.Si.** **(Penguji)**
   NIP: 19681124 199412 1 001

4. **Dr.rer.pol. Heri Kuswanto, M.Si.** **(Penguji)**
   NIP: 19820326 200312 1 004

Dekan
Fakultas Matematika, Komputasi dan Sains Data
Institut Teknologi Sepuluh Nopember

Prof. Dr. Basuki Widodo, M.Sc.
NIP: 19650605 198903 1 002

BOOSTING SUPPORT VECTOR MACHINE PADA DATA *MICROARRAY* YANG *IMBALANCE*

Nama Mahasiswa	**: Risky Frasetio Wahyu Pratama**
NRP	**: 06211650010002**
Pembimbing	**: Santi Wulan Purnami, M.Si., Ph.D**
Co Pembimbing	**: Dr. Santi Puteri Rahayu, M.Si.**

ABSTRAK

Data *microarray* memainkan peran penting dalam pengklasifikasian hampir semua jenis jaringan kanker. Permasalahan yang seringkali dihadapi dalam klasifikasi menggunakan data *microarray* adalah *high dimensional* data dan kelas *imbalance*. Masalah *high dimensional* data dapat diatasi dengan menggunakan seleksi fitur *Fast Correlated Based Filter*. Metode klasifikasi yang digunakan dalam penelitian ini yaitu *Support Vector Machines* (SVM) karena beberapa kelebihannya, namun SVM sangat sensitif terhadap kelas *imbalance*. SMOTE merupakan salah satu dalam penanganan data *imbalance* dengan cara mereplikasi pengamatan pada kelas minoritas. Metode ini seringkali bekerja baik namun terkadang juga terjadi masalah *overfitting*. Salah satu alternatif lain dalam meningkatkan performansi klasifikasi pada data *imbalance* yaitu *boosting*. Metode ini membangun suatu *classifier* akhir yang kuat dengan menggabungkan sekumpulan SVM sebagai *base classifier* selama proses iterasi, sehingga dapat meningkatkan performansi klasifikasi. Penelitian ini, bertujuan untuk mengkaji performansi dari SMOTEBoost-SVM jika dibandingkan dengan AdaBoost-SVM dalam melakukan klasifikasi pada data *microarray* dengan beberapa tingkatan rasio *imbalance* yang didesain dalam studi simulasi dan penerapan pada data publik *microarray*. Data publik yang digunakan yaitu data kanker colon dan data myeloma. Hasil analisis yang diperoleh yaitu secara umum, pada studi simulasi, semua *classifier* mengalami penurunan performansi g-mean seiring bertambahnya rasio kelas *imbalance*, namun SMOTEBoost-SVM cenderung unggul dan mengalami penurunan performansi lebih kecil (lebih stabil) dibandingkan AdaBoost-SVM, SMOTE-SVM dan SVM. Pada Penerapan data publik, SMOTEBoost SVM juga mengungguli ketiga metode lain berdasarkan ukuran g-mean dan *sensitivity*. Efek dari seleksi fitur juga dilihat dalam analisis dimana menggunakan fitur-fitur informatif hasil seleksi fitur, menghasilkan performansi yang lebih baik dibandingkan menggunakan seluruh fitur dalam klasifikasi.

Kata kunci: Data *Microarray*, Kelas *Imbalance*, FCBF, SMOTE, *Support Vector Machine*, AdaBoost, SMOTEBoost.

(halaman ini sengaja dikosongkan)

BOOSTING SUPPORT VECTOR MACHINE FOR IMBALANCED MICROARRAY DATA

By	**: Risky Frasetio Wahyu Pratama**
Student Identity Number	**: 06211650010002**
Supervisor	**: Santi Wulan Purnami, M.Si., Ph.D**
Co Supervisor	**: Dr. Santi Puteri Rahayu, M.Si.**

ABSTRACT

Microarray data plays an important role in the classification of almost all types of cancer tissue. The problems that often appear in the classification using microarray data are high-dimensional data and imbalanced class. The problem of high-dimensional data can be solved by using Fast Correlated Based Filter (FCBF) feature selection. In this paper, Support Vector Machine (SVM) classifier is used because of its advantages. However, SVM are sensitive with respect to imbalanced class. SMOTE is one of the prepocessing data methods in handling imbalanced class based on sampling approach by increasing the number of samples from the minority class. This method often works well but sometimes it might suffer from over-fitting problem. One other alternative approach in improving the performance of imbalanced data classification is boosting. This method constructs a powerful final classifier by combining a set of SVMs as base classifier during the iteration process. So, it can improve the classification performance. This study aims to see the performance of SMOTEBoost-SVM compared with AdaBoost-SVM in classifying microarray data with several levels of imbalance ratio designed in the simulation study and to apply classification process on public microarray datasets. Colon cancer and myeloma data are used in this study. The result showed that in the simulation study, all classifiers get the g-mean performance deacreasing as the ratio of the imbalanced class is increased, but SMOTEBoost-SVM tend to be superior. Its performance is decrease smaller (more stable) than AdaBoost-SVM, SMOTE-SVM and SVM. In the real data classification, SMOTEBoost-SVM outperforms the others with respect to g-mean and sensitivity metrics.The effect of feature selection is also checked in the analysis. Using informative features obtained in feature selection process gave the better performance than using all feature in the classification process by SVM.

Keywords: Microarray Data, Class Imbalance, FCBF, SMOTE, Support Vector Machine, AdaBoost, SMOTEBoost..

(halaman ini sengaja dikosongkan)

KATA PENGANTAR

Assalamu'alaikumWr. Wb.

Puji syukur kepada Allah S.W.T., atas rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan penyususnan Tesis dengan judul **"*BOOSTING SUPPORT VECTOR MACHINE* PADA DATA *MICROARRAY* YANG *IMBALANCE*"**. Selain itu tidak lupa sholawat serta salam penulis sampaikan kepada Nabi Muhammad SAW. Dalam menyusun Tesis ini, penulis ucapkan terimakasih kepada pihak-pihak yang membantu dalam menyelesaikan Tesis ini, khususnya kepada :

1. Bapak Dr. Suhartono, M.Sc selaku Ketua Jurusan Statistika FMKSD ITS, Surabaya
2. Bapak Dr.rer.pol. Heri Kuswanto, M.Si selaku Ketua Program Studi Magister Jurusan Statistika ITS Surabaya yang telah memberikan kemudahan dan motivasi kepada semua mahasiswa.
3. Ibu Santi Wulan Purnami, M.Si., Ph.D selaku dosen pembimbing yang telah banyak memberikan arahan, bimbingan, ilmu dan saran serta banyak hal baru yang telah diberikan kepada penulis dalam penyusunan Tesis ini.
4. Dr. Santi Puteri Rahayu, M.Si., selaku dosen co-pembimbing yang telah banyak memberikan arahan, bimbingan, ilmu dan motivasi kepada penulis dalam penyusunan Tesis ini.
5. Bapak Dr. Bambang Widjanarko Otok, M.Si selaku dosen penguji yang telah memberikan banyak kritik, saran dan arahan.
6. Bapak Dr.rer.pol. Heri Kuswanto, M.Si selaku dosen penguji yang telah memberikan banyak kritik, saran dan arahan.
7. Bapak dan Ibu dosen pengajar di Program Studi Magister Jurusan Statistika ITS Surabaya yang telah memberikan banyak ilmu selama perkuliahan di Program Studi Magister Jurusan Statistika ITS Surabaya

8. Kedua orang tua, Bapak Suwanto dan Ibu Mulyaningsih, serta Riska Frastiwi Wahyu Dwitama yang telah menjadi penyejuk hati dan tanpa henti memberkan dukungan dan doa kepada penulis.
9. Teman terbaik, Alvita Rachma Devi atas segala dukungan dan doa dalam penyelesaian Tesis ini.
10. Teman-teman angkatan S2 angkatan ganjil 2016/2017 yang telah berbagi suka duka selama perkuliahan di Program Studi Magister Jurusan Statistika ITS Surabaya
11. Semua pihak yang tidak dapat disebutkan satu-persatu yang telah membantu hingga Tesis ini dapat terselesaikan dengan baik.

Penulis menyadari sepenuhnya bahwa Tesis ini masih jauh dari sempurna, oleh karena itu segala kritik dan saran yang sifatnya membangun selalu penulis harapkan. Semoga Tesis ini dapat bermanfaat bagi penulis khususnya dan bagi semua yang membutuhkan umumnya. Akhir kata, semoga Allah SWT selalu melimpahkan rahmat serta hidayah-Nya kepada kita semua, Amin amin ya robbal ‘alamiin.

Surabaya, July 2018

Penulis

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR TABEL xv
DAFTAR GAMBAR xvii
DAFTAR LAMPIRAN xix
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 7
1.3 Tujuan Penelitian 7
1.4 Manfaat Penelitian 7
1.5 Batasan Masalah 7
BAB 2 TINJAUAN PUSTAKA 9
2.1 Seleksi Fitur Menggunakan FCBF 9
2.2 *Support Vector Machine* 13
2.2.1 SVM *Linearly Separable* 15
2.2.2 SVM *Linearly Non Separable* 19
2.2.3 SVM *Non Linear* 22
2.3 SMOTE 24
2.4 *Boosting* 25
2.5 SMOTEBoost 28
2.6 *K-Fold Cross Validation* 30
2.7 Evaluasi Performansi 31
2.8 Ekspresi Gen dan Data *Microarray* 32
BAB 3 METODOLOGI PENELITIAN 35

3.1 Tahapan Dalam Membangun AdaBoost-SVM 35
3.2 Tahapan Dalam Membangun SMOTEBoost-SVM 38
3.3 Tahapan pada Studi Simulasi 41
3.4 Tahapan pada Klasifikasi Data Publik *Microarray* 43
BAB 4 HASIL DAN PEMBAHASAN 49
4.1 Algoritma AdaBoost-SVM 49
4.2 Algoritma SMOTEBoost-SVM 51
4.3 Studi Simulasi 52
4.3.1 Karakteristik Data Simulasi 53
4.3.2 Klasifikasi pada Data Skenario Rasio *Imbalance* 55
4.3.3 Klasifikasi pada Data Rasio *Imbalance* 2 58
4.3.4 Klasifikasi pada Data Rasio *Imbalance* 5 60
4.3.5 Klasifikasi pada Data Rasio *Imbalance* 10 62
4.3.6 Klasifikasi pada Data Rasio *Imbalance* 15 65
4.3.7 Perbandingan Performansi Klasifikasi 67
4.4 Klasifikasi pada Data *Microarray* 69
4.4.1 Klasifikasi Menggunakan SVM 72
4.4.2 Klasifikasi Menggunakan SMOTE-SVM 76
4.4.3 Klasifikasi Menggunakan *Boosting*-SVM 79
4.4.3.1 Klasifikasi Menggunakan AdaBoost-SVM 79
4.4.3.2 Klasifikasi Menggunakan SMOTEBoost-SVM 82
4.4.4 Perbandingan Performansi Klasifikasi 85
BAB 5 KESIMPULAN DAN SARAN 89
5.1 Kesimpulan 89
5.2 Saran 90
DAFTAR PUSTAKA 91
LAMPIRAN 97

DAFTAR TABEL

Tabel 2.1 Data Ilustrasi Seleksi Fitur 11
Tabel 2.2 Ilustrasi Nilai *Pairwise SU* 13
Tabel 2.3 Matriks Konfusi 31
Tabel 3.1 Deskripsi Data 40
Tabel 3.2 Ilustrasi Proses Validasi 45
Tabel 4.1 Nilai Performansi Pada Klasifikasi Data Rasio *Imbalance* 1 pada Setiap Parameter 56
Tabel 4.2 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 1 Menggunakan Parameter Optimum 57
Tabel 4.3 Nilai Performansi Pada Klasifikasi Data Rasio *Imbalance* 2 pada Setiap Parameter 58
Tabel 4.4 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 2 Menggunakan Parameter Optimum 59
Tabel 4.5 Nilai Performansi Pada Klasifikasi Data Rasio *Imbalance* 5 pada Setiap Parameter 60
Tabel 4.6 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 5 Menggunakan Parameter Optimum 62
Tabel 4.7 Nilai Performansi Pada Klasifikasi Data Rasio *Imbalance* 10 pada Setiap Parameter 63
Tabel 4.8 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 10 Menggunakan Parameter Optimum 65
Tabel 4.9 Nilai Performansi Pada Klasifikasi Data Rasio *Imbalance* 15 pada Setiap Parameter 65
Tabel 4.10 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 15 Menggunakan Parameter Optimum 67
Tabel 4.11 Jumlah Fitur Sebelum dan Sesudah Seleksi Fitur 69
Tabel 4.12 Rangkuman Performansi Klasifikasi SVM pada Kedua Data Menggunakan Seluruh Fitur dan Beberapa Fitur Informatif 75
Tabel 4.13 Distribusi Kelas Pengamatan Sebelum dan Setelah Dilakukan

Seleksi Fitur 76
Tabel 4.14 Rangkuman Performansi Klasifikasi Data Kanker Colon Menggunakan SMOTE-SVM pada Beberapa Parameter 77
Tabel 4.15 Rangkuman Performansi Klasifikasi Data Kanker Myeloma Menggunakan SMOTE-SVM pada Beberapa Parameter 78
Tabel 4.16 Rangkuman Performansi Klasifikasi SMOTE-SVM pada Beberapa Parameter Optimum 78
Tabel 4.17 Rangkuman Performansi *G-mean* pada Klasifikasi Kedua Data Menggunakan AdaBoost-SVM dengan Beberapa Parameter 80
Tabel 4.18 Rangkuman Performansi Klasifikasi AdaBoost-SVM pada Parameter Optimum 82
Tabel 4.19 Rangkuman Performansi *G-mean* pada Klasifikasi Kedua Data Menggunakan SMOTEBoost-SVM dengan Beberapa Parameter 83
Tabel 4.20 Rangkuman Performansi Klasifikasi SMOTEBoost-SVM pada Parameter Optimum 85
Tabel 4.21 Perbandingan Performansi Klasifikasi Data *Microarray* 87

DAFTAR GAMBAR

Gambar 2.1 Algoritma FCBF 11
Gambar 2.2 Ilustrasi *Hyperplane* yang Mungkin pada Suatu Pengklasifikasian 14
Gambar 2.3 Klasifikasi SVM 15
Gambar 2.4 *Hyperplane* dan *Margin* dalam Kasus *Linearly Separable* 16
Gambar 2.5 *Hyperplane* dan *Margin* dalam Kasus *Linearly Non Separable* 20
Gambar 2.6 Pemetaan Ruang Fitur Data Input Dua Dimensi ke Ruang Fitur Tiga Dimensi 22
Gambar 2.7 Ilustrasi SMOTE 25
Gambar 2.8 Algoritma AdaBoost 27
Gambar 2.9 Algoritma SMOTEBoost dengan Prosedur Iterasi AdaBoost 30
Gambar 2.10 Proses dalam Memperoleh Data *Microarray* 33
Gambar 3.1 Tahapan AdaBoost-SVM 37
Gambar 3.2 Tahapan SMOTEBoost-SVM 40
Gambar 3.3 Tahapan Penelitian 47
Gambar 4.1 Persentase Distribusi Kelas pada Tiap Skenario Data 54
Gambar 4.2 Pola Persebaran Kelas pada Tiap Data Bangkitan 55
Gambar 4.3 *Boxplot* Performansi Klasifikasi Data Rasio *Balance* Menggunakan Parameter Optimum 57
Gambar 4.4 *Boxplot* Performansi Klasifikasi Data Rasio *Imbalance* 2 Menggunakan Parameter Optimum 59
Gambar 4.5 *Boxplot* Performansi Klasifikasi Data Rasio *Imbalance* 5 Menggunakan Parameter Optimum 61
Gambar 4.6 *Boxplot* Performansi Klasifikasi Data Rasio *Imbalance* 10 Menggunakan Parameter Optimum 64
Gambar 4.7 *Boxplot* Performansi Klasifikasi Data Rasio *Imbalance* 15

Menggunakan Parameter Optimum 66
Gambar 4.8 Perbandingan Performansi Model pada Data Simulasi..... 67
Gambar 4.9 Distribusi Kelas pada Data *Microarray*..... 69
Gambar 4.10 Pola Persebaran Kelas Pengamatan Data Kanker Colon dan Myeloma pada Beberapa Fitur Informatif..... 69
Gambar 4.11 Boxplot Nilai-Nilai Ekspresi Gen pada Beberapa Fitur Informatif pada Data Kanker Colon..... 70
Gambar 4.12 Boxplot Nilai-Nilai Ekspresi Gen pada Beberapa Fitur Informatif pada Data Myeloma..... 71
Gambar 4.13 Performansi G-mean pada Klasifikasi SVM pada Data Kanker Colon Menggunakan Beberapa nilai C dan γ Kernel Radial.... 72
Gambar 4.14 Performansi G-mean pada Klasifikasi SVM pada Data Kanker Colon Menggunakan Beberapa nilai C Kernel Linier..... 73
Gambar 4.15 Performansi G-mean pada Klasifikasi SVM pada Data Myeloma Menggunakan Beberapa nilai C dan γ Kernel Radial..... 73
Gambar 4.16 Performansi G-mean pada Klasifikasi SVM pada Data Kanker Myeloma Menggunakan Beberapa nilai C Kernel Linier..... 74
Gambar 4.17 Plot Nilai Performansi pada beberapa Iterasi Maksimal AdaBoost-SVM pada Klasifikasi Data Kanker Colon..... 81
Gambar 4.18 Plot Nilai Performansi pada beberapa Iterasi Maksimal AdaBoost-SVM pada Klasifikasi Data Myeloma..... 81
Gambar 4.19 Plot Nilai Performansi pada beberapa Iterasi Maksimal SMOTEBoost-SVM pada Klasifikasi Data Kanker Colon..... 84
Gambar 4.20 Plot Nilai Performansi pada beberapa Iterasi Maksimal SMOTEBoost-SVM pada Klasifikasi Data Myeloma..... 84
Gambar 4.21 *Boxplot* Nilai-Nilai G-mean pada Beberapa Parameter 87

DAFTAR LAMPIRAN

Lampiran 1	*Syntax* Fungsi AdaBoost-SVM	97
Lampiran 2	*Syntax* Fungsi SMOTEBoost-SVM	99
Lampiran 3	*Syntax* Klasifikasi Data Riil Menggunakan SVM	101
Lampiran 4	*Syntax* Klasifikasi Data Riil Menggunakan SMOTE-SVM	103
Lampiran 5	*Syntax* Klasifikasi Data Riil Menggunakan AdaBoost-SVM ..	107
Lampiran 6	*Syntax* Klasifikasi Data Riil Menggunakan SMOTEBoost-SVM	110
Lampiran 7	*Syntax* pada Studi Simulasi....	112
Lampiran 8	Hasil Seleksi Fitur Menggunakan FCBF pada Data Riil.....	126
Lampiran 9	Performansi G-mean pada Klasifikasi Data Riil Menggunakan AdaBoost-SVM....	128
Lampiran 10	Performansi G-mean pada Klasifikasi Data Riil Menggunakan SMOTEBoost-SVM....	130
Lampiran 11	Performansi G-mean pada Klasifikasi Data Riil Menggunakan SVM....	132
Lampiran 12	Performansi G-mean pada Klasifikasi Data Riil Menggunakan SMOTE-SVM....	134
Lampiran 13	Ilustrasi Proses Klasifikasi AdaBoost-SVM.....	136
Lampiran 14	Ilustrasi Proses Klasifikasi SMOTEBoost-SVM.....	146

(halaman ini sengaja dikosongkan)

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dalam studi mengenai penyakit kanker, data *microarray* memainkan peran penting dalam pendeteksian dan pengklasifikasian hampir semua jenis jaringan kanker. Data *microarray* berisi suatu matrik dengan ribuan kolom dan ratusan baris, dimana setiap kolom dan baris masing-masing menyajikan gen dan sampel (Saberkari dkk, 2014). Ekspresi gen yang dihasilkan dari teknologi *microarray* membawa informasi gen-gen yang kemudian dicocokkan dengan suatu kondisi penyakit tertentu termasuk kanker, baik dalam biologi dan pengobatan klinik pada tingkat molekuler (Trevino dkk, 2007). Sebagaimana telah diketahui bahwa kanker merupakan suatu penyakit genetik yang paling mematikan. Kanker terjadi melalui mutasi genetik atau perubahan epigenetik yang membawa perubahan pada profil ekspresi gen dari sel-sel kanker. Akibatnya, teknologi *microarray* saat ini banyak dimanfaatkan untuk mengidentifikasi perubahan gen yang memainkan peran pada suatu kanker tertentu, dan menemukan *biomarker* terbaru untuk diagnosis klinis (Slonim, 2002). Salah satu tantangan dalam mengolah data *microarray* adalah masalah dimensi data. Mengingat data *microarray* merupakan data dengan dimensi yang tinggi, dengan jumlah fitur yang sangat besar, maka terlebih dahulu perlu dilakukan seleksi fitur (Wang dkk, 2011). Tujuan seleksi fitur adalah untuk mengurangi tingkat kompleksitas dari suatu algoritma klasifikasi, dan mengetahui fitur-fitur yang paling berpengaruh terhadap tingkatan kelas (Abraham dkk, 2009). Seleksi fitur juga dapat mengimprovisasi akurasi klasifikasi dengan membuang fitur-fitur yang redundan dan tidak relevan (Purnami dkk, 2017). Salah satu metode seleksi fitur yang belakangan ini banyak digunakan yaitu FCBF (*Fast Corelation Based Filter*). FCBF merupakan algoritma seleksi fitur baru yang terbukti bekerja secara cepat dan mampu memilih fitur yang terbaik serta mempertimbangkan kecepatan waktu (Yu dan Liu, 2003).

Klasifikasi merupakan salah satu teknik yang paling populer di dalam *data mining* atau *machine learning*.. Tujuan dari klasifikasi adalah menemukan suatu fungsi keputusan yang secara akurat memprediksi kelas dari data *testing* yang berasal dari fungsi distribusi yang sama dengan data untuk *training*. Beberapa metode klasifikasi yang umum digunakan antara lain: Regresi Logistik, Analisis Diskriminan (LDA), *Artificial Neural Network* (ANN), dan *Support Vector Machine* (SVM). Salah satu contoh penerapan klasifikasi yaitu pada diagnosa klinis. Pada diagnosa medis khususnya mengenai penyakit kanker, data *microarray* banyak digunakan. Beberapa penelitian mengenai klasifikasi kanker molekuler menggunakan data *microarray* diantaranya: Klasifikasi kanker leukimia oleh Batista (2004) dan Guyon (2002), klasifikasi kanker otak oleh Pomerroy (2002), klasifikasi kanker paru oleh Bhattacharjee (2001), klasifikasi kanker kelenjar getah bening oleh Shipp (2002).

Support Vector Machine (SVM) adalah suatu metode klasifikasi yang awalnya diusulkan oleh Vapnik (1995). SVM telah diaplikasikan secara sukses dalam banyak masalah klasifikasi dan regresi di berbagai bidang. Prinsipnya yaitu menemukan sekumpulan pemisah optimal (*hyperplane*) dari data klasifikasi yang dilatih dengan suatu algoritma sehingga dapat memisahkan dataset menjadi dua atau lebih kelas yang berbeda yang dapat membantu memprediksi kategori dari data baru (Huang dkk, 2014). Keuntungan menggunakan SVM antara lain memiliki *background* matematika yang solid, kemampuan generalisasi yang tinggi dan kemampuan dalam menemukan solusi klasifikasi yang global dan non linier (Batuwita dan Palade, 2013; Purnami dkk, 2015). Beberapa contoh penggunan SVM dalam masalah klasifikasi antara lain di bidang ekonomi yaitu klasifikasi risiko kredit oleh Yu (2008), prediksi *financial distress* oleh Sun dan Li (2012), dll. Contoh penggunaan pada *pattern recognition* yaitu pada deteksi wajah oleh Demidova (2016). SVM juga telah menunjukan hasil yang menjanjikan dalam beberapa publikasi masalah klasifikasi biologi termasuk dalam mengevaluasi ekspresi gen pada data *microarray*. Brown (2000) melakukan analisis data ekspresi gen *microarray* dari *database Munich Information Center for Protein Sequences Yeast Genome* untuk mengklasifikasikan 5 kategori kelas

gen. Brown (2000) membandingkan *classifer* SVM, LDA dan *Decision Tree.* Hasilnya menunjukkan SVM superior dibandingkan *classifier* yang lain. Penelitian lain mengenai penggunaan SVM pada klasifikasi data *microarray* dilakukan oleh Vanitha (2015). Pada penelitian tersebut, peneliti melakukan dua analisis terhadap data kanker *colon* oleh Alon (1999) dan data kanker *lymphoma* oleh Alizadeh (2000). Hasil penelitian menunjukan bahwa performansi SVM dengan seleksi fitur lebih baik dibandingkan *classifier* KNN dan ANN dengan akurasi 67,7% pada data kanker *colon* dan 97,7% pada data kanker *lymphoma*.

Dalam aplikasi di lapangan, banyak sekali *dataset* yang *imbalance,* yang mana kelas (kasus) yang ingin diteliti cenderung terdistribusi lebih banyak atau lebih sedikit. Salah satu contohnya yaitu pendiagnosaan medis suatu penyakit yang jarang (langka) dimana kelas positif yang akan diteliti adalah kelas yang jarang (minor). Dalam kasus tersebut, akurasi klasifikasi pada kelas mayor akan cenderung tinggi sedangkan akurasi klasifikasi kelas minor akan cenderung rendah, sehingga biaya (kerugian) akan besar saat sebuah *classifier* salah mengklasifikasikan pengamatan kelas yang jarang (minor) (Liu, 2016; Sain dan Purnami, 2015). Belakangan ini permasalahan kelas *imbalance* telah dikenali sebagai masalah yang krusial dalam *machine learning* dan *data mining.* Salah satu contoh masalah kelas *imbalance* yaitu pada masalah klasifikasi yang menggunakan data *microarray*.

Hampir semua *classifier* termasuk SVM mengasumsikan sebuah pembagian yang rata antar kelas-kelas pengamatan. Walaupun SVM seringkali bekerja secara efektif pada *dataset* yang *balance*, namun SVM bisa menghasilkan hasil yang kurang optimal pada *dataset* yang *imbalance* (Batuwita dan Palade, 2013). Veropoulos (1999), Wu dan Chang (2003) dan Akbani (2004) telah melakukan studi mengenai masalah data *imbalance* pada SVM dan menemukan beberapa alasan SVM sensitif terhadap kelas *imbalance*. Alasan pertama yaitu masalah optimasi *soft margin,* telah diidentifikasi bahwa pemisah *hyperplane* yang dikembangkan dengan suatu data set yang *imbalance* akan condong (miring) ke arah kelas yang minor. Alasan kedua yaitu saat melakukan *training* pada data yang *imbalance*, rasio diantara *support vector* yang positif dan negatif juga

menjadi tidak *balance*. Wu dan Chang (2003) membuat suatu hipotesis bahwa sebagai dampak atas hasil *support vector* yang *imbalance* ini, pengamatan-pengamatan yang berada di sekitar batas akan cenderung didominasi oleh *support vector* yang negatif dan oleh karenanya fungsi pemisah akan cenderung mengklasifikasikan sebuah titik dekat batas sebagai kelas negatif.

Terdapat beberapa cara dalam menangani permasalahan kelas *imbalance* pada klasifikasi menggunakan SVM. Batuwita dan Palade (2013) pada publikasinya menyebutkan terdapat dua dasar utama metode-metode dalam penanganan masalah klasifikasi data *imbalance* pada *classifier* SVM, diantaranya: metode yang berdasar pada *preprocessing* data (metode *resampling* data dan *ensemble learning*), dan metode yang berdasar pada algoritma yang terbagi atas metode *different cost error*, zSVM, pemodifikasian kernel, *active learning*, dan *fuzzy* SVM. Prinsip kerja pendekatan berbasis *sampling* dalam penanganan kelas *imbalance* yaitu memodifikasi distribusi data *training* sehingga kedua kelas data baik kelas positif (minor) dan kelas negatif (mayor) dipresentasikan dengan baik (seimbang) di dalam data *training*. Secara umum pendekatan berbasis *sampling* dibagi menjadi dua yaitu metode *oversampling* dan *undersampling*. Metode *undersampling* menyeimbangkan data dengan cara menghapus beberapa pengamatan pada kelas mayor hingga keseimbangan data *training* yang diinginkan tercapai. Namun kelemahan metode ini yaitu kemungkinan hilangnya informasi yang berasosiasi dengan penghapusan pengamatan dari data *training* (Batista, 2004). Pendekatan berbasis *sampling* yang kedua yaitu *oversampling*. Metode ini bekerja untuk menyeimbangkan data *training* dengan cara meningkatkan jumlah data pada kelas minor. Salah satu metode *oversampling* yang paling efektif yaitu SMOTE (*Synthetic Minority Oversampling Technique*).

Algoritma SMOTE diusulkan oleh Chawla dkk (2002). Algoritma ini cenderung bekerja membuat data *synthetic* pada kelas minor dengan melakukan proses ekstrapolasi diantara pengamatan-pengamatan kelas minor yang sudah ada dibandingkan menduplikasi pengamatan asal dengan cara yang simpel. Proses ekstrapolasi dilakukan berdasarkan k-tetangga terdekatnya. Penggunaan SMOTE dalam beberapa publikasi mengenai penanganan kelas *imbalance* dinilai cukup

berhasil. Salah satu penelitian mengenasi SMOTE dilakukan oleh Purnami dan Trapsilasiwi (2015). Dalam penelitian tersebut digunakan algoritma SMOTE dan *Least Square* SVM pada kasus klasifikasi kanker multi kelas. Purnami dan Trapsilasiwi menyimpulkan bahwa penambahan algoritma SMOTE pada LS SVM memberikan hasil yang lebih baik dibandingkan LS SVM standar. Namun kelemahan algoritma SMOTE ini yaitu seringkali mengalami *overfitting* (Drummond dan Holte, 2003). Penelitian lain yang berbasis pada *sampling* dilakukan oleh Sain dan Purnami (2015) dengan mengkombinasikan *sampling* antara SMOTE dan *Tomek link* dengan *classifier* SVM, hasilnya *combine sampling* memberikan nilai akurasi yang lebih baik dibandingkan *sampling* SMOTE dan *Tomek link*. Namun, Sain dan Purnami (2015) juga menyebutkan bahwa dalam beberapa kasus *imbalance* yang ekstrim *combine sampling* dapat tampil kurang baik.

Salah satu alternatif lain dalam meningkatkan akurasi kelas *imbalance* adalah dengan menggunakan metode *ensemble*. Metode *ensemble* pada prinsipnya mengkombinasikan sekumpulan *classifier* yang dilatih dengan tujuan untuk membuat model klasifikasi (*classifier*) campuran yang terimprovisasi sehingga membuat *classifier* gabungan yang terbentuk lebih akurat dari pada *classifier* asalnya dalam melakukan suatu pengklasifikasian (Han dkk, 2012). Salah satu metode *ensemble* yang sangat popular yaitu *boosting*. Prinsip kerja *boosting* yaitu mempekerjakan sekumpulan *classifier* yang dilatih secara iteratif. *AdaBoost* yang diperkenalkan oleh Freund dan Schapire (1995) adalah salah satu algoritma *boosting* yang paling populer. Kesuksesan penerapan *AdaBoost* salah satunya yaitu dapat meluaskan *margin* yang mana dapat meningkatkan kemampuan generalisasi yang lebih baik. *AdaBoost* pada prinsipnya membentuk satu *classifier* yang kuat dengan mengkombinasikan sekumpulan *classifier*. *AdaBoost* mempertahankan sekumpulan bobot pengamatan pada saat *training* pengamatan dan secara adaptif menyesuaikan (*updating*) bobot-bobot ini pada akhir tiap iterasi *boosting*. Bobot-bobot dari pengamatan yang salah terklasifikasikan pada saat *training* akan dinaikkan sementara bobot-bobot pengamatan yang terklasifikasikan dengan benar akan diturunkan nilainya (Li dkk, 2008). Dengan

kata lain, *AdaBoost* memaksa suatu *classifier* untuk memberi perhatian yang lebih pada pengamatan yang salah (sulit) diklasifikasikan (Garcia dan Lozano, 2007). Banyak penelitian telah dipublikasikan tentang penerapan *boosting* menggunakan *decision tree*, regresi logistik, *neural network*, dan SVM sebagai komponen *classifier*. Penelitian-penelitian tersebut menunjukkan performa generalisasi yang baik. Salah satu penelitian mengenai Adaboost dengan *base classifier* SVM dilakukan oleh (Li dkk, 2008). Pada penelitiannya, AdaBoost dengan SVM radial sebagai komponen *classifier* menghasilkan performansi yang lebih baik dibandingkan SVM standar pada kelima *dataset* yang digunakan.

Salah satu algoritma *boosting* lainnya yang terbilang sukses dalam penerapaannya akhir-akhir ini yaitu SMOTEBoost yang diusulkan oleh Chawla dkk (2003). SMOTEboost memodifikasi algoritma AdaBoost dengan menambahkan algoritma SMOTE di tiap iterasi *boosting*. Penambahan SMOTE di tiap iterasi *boosting* bertujuan untuk menambah probabilitas terpilihnya sanpel-sampel yang sulit diklasifikasikan yang berasal dari kelas minoritas. Pada penelitian tersebut Chawla (2003) menggunakan algoritma RIPPER sebagai komponen *classifier*. Hasil penelitiannya menunjukkan bahwa SMOTEBoost menghasilkan performansi yang lebih baik dibandingkan AdaBoost dalam klasifikasi keempat *dataset imbalance* yang digunakan.

Berdasarkan kekurangan dan kelebihan metode yang telah dibahas sebelumnya, maka dalam penelitian ini, peneliti memilih menggunakan algoritma *boosting* dalam melakukan klasifikasi data *microarray* yang *imbalance*. Dua algoritma *boosting* yang digunakan yakni AdaBoost dan SMOTEBoost. Algoritma SVM digunakan sebagai komponen *classifier*. Sebelumnya dilakukan seleksi fitur menggunakan metode *filter* FCBF untuk mereduksi dimensi data dalam hal mengurangi kompleksitas algoritma dan mencari fitur-fitur terbaik. Pada penelitian ini dilakukan studi simulasi untuk melihat performansi dari metode-metode yang digunakan dalam klasifikasi data *microarray* dengan beberapa tingkatan rasio *imbalance* yang berbeda-beda.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, terdapat dugaan bahwa algoritma *boosting* SMOTEBoost akan dapat meningkatkan performansi klasifikasi menggunakan data *microarray* sehingga rumusan permasalahan pada penelitian ini yaitu bagaimana performansi SMOTEBoost-SVM jika dibandingkan dengan algoritma AdaBoost-SVM, SMOTE-SVM dan SVM dalam melakukan klasifikasi pada data *microarray* yang *imbalance*. Untuk itu, dilakukan studi simulasi dengan desain skenario yang berfokus pada rasio dari kelas *imbalance* dan penerapan pada data publik *microarray*.

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini yaitu

1. Mengkaji performansi algoritma SMOTEBoost-SVM dan AdaBoost-SVM dalam melakukan klasifikasi pada data *microarray* dengan beberapa tingkat rasio kelas *imbalance* yang berbeda menggunakan studi simulasi.
2. Membandingkan algoritma AdaBoost-SVM dan SMOTEBoost-SVM dalam melakukan klasifikasi pada data publik *microarray*.

1.4 Manfaat Penelitian

Adapun manfaat dari penelitian ini yaitu

1. Memberikan informasi mengenai penerapan *boosting* guna meningkatkan performansi *classifier* SVM dalam pengklasifikasian data *microarray* yang *imbalance*.
2. Menambah keilmuan statistika di bidang *machine learning* khususnya pada teknik klasifikasi pada data *imbalance*.

1.5 Batasan Masalah

Pada penelitian ini penanganan data *imbalance* dilakukan dengan pendekatan *preprocessing* dimana Adaboost dan SMOTEBoost digunakan dalam penanganan kelas *imbalance* berdasarkan kelebihan-kelebihan yang telah diuraikan pada latar belakang. Algoritma klasifikasi yang digunakan yaitu SVM

dengan fungsi Kernel yang digunakan yaitu Kernel linier dan Kernel *Radial Basis Function*. Permasalahan klasifikasi yang akan diteliti pada penelitian ini adalah permasalahan klasifikasi biner. Batasan yang lain yakni Klasifikasi menggunakan AdaBoost-SVM dan SMOTEBoost-SVM hanya dilakukan menggunakan fitur yang informatif sehingga efek dari seleksi fitur tidak dapat di analisis pada saat melakukan klasifikasi menggunakan kedua metode *boosting* tersebut. Selain itu, studi simulasi yang dilakukan juga hanya berfokus pada skenario rasio kelas *imbalance*.

BAB 2
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai landasan teori yang mendukung metode-metode yang akan digunakan dalam penelitian ini diantaranya *Fast Correlation Based Filter* (FCBF), *Support Vector Machine* (SVM), *Synthetic Minority Oversampling Technique* (SMOTE), AdaBoost, SMOTEBoost, *k-fold cross validation*, evaluasi performansi, dan tinjauan tentang data *microarray*.

2.1 Seleksi Fitur Menggunakan Fast Correlation Based Filter (FCBF)

Seleksi fitur adalah proses memilih fitur untuk digunakan dalam proses klasifikasi atau klastering. Tujuan dari seleksi fitur adalah untuk mengurangi tingkat kompleksitas dari sebuah algoritma klasifikasi, meningkatkan akurasi dari algoritma klasifikasi tersebut dan mampu mengetahui fitur-fitur yang paling berpengaruh terhadap tingkat kelas. Pendekatan yang ideal untuk fitur seleksi yaitu mencoba semua subset-subset yang mungkin dari fitur sebagai input untuk suatu algoritma data mining, dan kemudian mengambil subset yang memproduksi hasil yang terbaik. Menurut Tan dkk (2006) terdapat tiga pendekatan standar untuk seleksi fitur yaitu metode *filter*, *embedded*, dan *wrapper*. Pada pendekatan *filter*, fitur dipilih sebelum algoritma data *mining* dijalankan menggunakan beberapa pendekatan yang independen dengan penugasan algoritma data *mining*.

FCBF merupakan salah satu algoritma seleksi fitur yang bersifat multivariat dan mengukur kelas fitur dan korelasi antara fitur-fitur (Bolon dkk, 2015). FCBF didesain untuk mengatasi kasus data dengan dimensi tinggi yang diharapkan mampu untuk mengurangi atau menyeleksi fitur yang tidak penting dan fitur yang berlebihan. Pendekatan *linear correlation coefficient* untuk setiap variabel (X,Y) dirumuskan pada Persamaan (2.1)

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(\mathrm{y}_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(\mathrm{y}_i - \bar{y})^2}} \qquad (2.1)$$

$\bar{x}_i$ adalah rata-rata dari X dan $\bar{y}_i$ adalah rata-rata dari Y dimana $i = 1,2,3,\dots,n$ serta rentang nilai r berbeda antara -1 dan 1. Jika X dan Y memiliki korelasi maka nilai r adalah 1 atau -1. Jika tidak berkorelasi maka nilai r adalah 0. Namun jika hubungan antar fitur tidak linier maka korelasi linier tidak dapat digunakan. Ukuran lain yang dapat digunakan yakni *entropy*. *Entropy* dari variabel X didefinisikan pada Persamaan (2.2).

$$H(X) = -\sum_{i}^{n} P(x_i)\log_2(P(x_i)),\ i = 1,2,...,\mathrm{n} \qquad (2.2)$$

Entropy dari variabel X jika diketahui variabel Y didefinisikan pada Persamaan (2.3).

$$H(X \mid Y) = -\sum_{i}^{n} P(\mathrm{y}_i)\sum_{i}^{n} P(x_i \mid \mathrm{y}_i)\log_2(P(x_i \mid \mathrm{y}_i)),\ i = 1,2,...,\mathrm{n} \qquad (2.3)$$

$P(x_i)$ adalah *posterior probabilities* untuk semua nilai X dan $P(x_i|y_i)$ adalah *posterior probabilities* dari X jika Y diketahui. Dari *entropy* tersebut dapat diperoleh *Information Gain* sebagai berikut:

$$IG(X \mid Y) = H(X) - H(X \mid Y) \qquad (2.4)$$

Untuk mengukur korelasi antar fitur, maka digunakan *symmetrical uncertainty*. Nilai *symmetrical uncertainty* berkisar pada rentang 0 sampai dengan 1. *Symmetrical uncertainty* dirumuskan sebagai berikut.

$$SU(X,Y) = 2\frac{IG(X \mid Y)}{H(X) + H(Y)} \qquad (2.5)$$

Algoritma FCBF secara umum ditunjukan pada Gambar 2.1.

```
input :   S(F1, F2,...,FN, C)   // training dataset
          δ                     // nilai threshold yang telah ditentukan
output:   Sbest                 // sekumpulan feature optimal
1   begin
2     for i =1 to N do begin
3         calculate SUi,c or Fi;
4     if ( SUi,c ≥ δ )
5         append Fi to S'list;
6     end;
7     order S'list in descending SUi,c value;
8     Fp = getFirstElement(S'list);
9     do begin
10        Fq = getNextElement(S'list, Fq);
11        if (Fq <>NULL)
12           do begin
13              F'q = Fq;
14              if (SUp,q ≥ SUq,c)
15                 remove Fq from S'list;
16                 Fq = getNextElement(S'list, F'p);
17              else Fq = getNextElement(S'list, Fq);
18           end until (Fq = NULL);
19        Fp = getNextElement(S'list, Fp);
20     end until (Fp ==NULL)
21  Sbest =S'list;
22  end;
```

Gambar 2.1. Algoritma FCBF (Yu dan Liu, 2003)

Selanjutnya akan diberikan sebuah ilustrasi proses seleksi fitur FCBF pada data keputusan bermain melalui Tabel 2. Dimana kelas label dari data berikut yakni keputusan bermain.

Tabel 2.1 Data Ilustrasi Seleksi Fitur (Santosa, 2007)

Angin	Cuaca	Temperatur	Kelembapan	Keputusan
Kecil	Cerah	Panas	Tinggi	Tidak
Besar	Cerah	Panas	Tinggi	Tidak
Kecil	Mendung	Panas	Tinggi	Main
Kecil	Hujan	Sedang	Tinggi	Main
Kecil	Hujan	Dingin	Normal	Main
Besar	Hujan	Dingin	Normal	Tidak
Besar	Mendung	Dingin	Normal	Main
Kecil	Cerah	Sedang	Tinggi	Tidak
Kecil	Cerah	Dingin	Normal	Main
Kecil	Hujan	Sedang	Normal	Main
Besar	Cerah	Sedang	Normal	Main
Besar	Mendung	Sedang	Tinggi	Main
Kecil	Mendung	Panas	Normal	Main
Besar	Hujan	Sedang	Tinggi	Tidak

Proses pertama dalam seleksi fitur FCBF yaitu mencari fitur-fitur yang relevan terhadap kelas berdasarkan nilai *SU* setiap fitur terhadap kelas. Sebagai contoh akan dihitung nilai *SU*(keputusan,angin). Perhitungan nilai *entropy* untuk kelas keputusan dan angin sebagai berikut

$$H(\text{keputusan}) = -\left(\frac{9}{14}\log_2\left(\frac{9}{14}\right)+\frac{5}{14}\log_2\left(\frac{5}{14}\right)\right) = 0{,}9403$$

$$H(\text{angin}) = -\left(\frac{6}{14}\log_2\left(\frac{6}{14}\right)+\frac{8}{14}\log_2\left(\frac{8}{14}\right)\right) = 0{,}985$$

Kemudian perhitungan $H(\textit{keputusan}\,|\,\textit{angin})$ adalah sebagai berikut

$$= -\left(\frac{8}{14}\left(\frac{6}{8}\log_2\left(\frac{6}{8}\right)+\frac{2}{8}\log_2\left(\frac{2}{8}\right)\right)+\frac{6}{14}\left(\frac{3}{6}\log_2\left(\frac{3}{6}\right)+\frac{3}{6}\log_2\left(\frac{3}{6}\right)\right)\right)$$

$$= -\left(\frac{8}{14}(-0{,}811)+\frac{6}{14}(-1)\right) = 0{,}892$$

sehingga nilai *information gain* antara fitur angin dan keputusan diperoleh

$$IG(\text{keputusan}\,|\,\text{angin}) = H(\text{keputusan}) - H(\text{keputusan}\,|\,\text{angin}) = 0{,}048$$

Nilai *Symmetrical Uncertainty* yang menyatakan hubungan fitur keputusan dan angin yaitu

$$SU(\text{keputusan, angin}) = 2\left[\frac{0{,}048}{0{,}985+0{,}9403}\right] = 0{,}05$$

Dengan cara yang sama dihitung nilai *SU* untuk setiap fitur terhadap kelas diperoleh nilai-nilai yang diurutkan dari yang tertinggi sebagai berikut

$$SU(\text{keputusan, cuaca}) = 0{,}196$$

$$SU(\text{keputusan, kelembapan}) = 0{,}157$$

$$SU(\text{keputusan, temperatur}) = 0{,}023$$

Misalkan diberikan suatu nilai *threshold* = 0,05 maka fitur-fitur yang relevan terhadap kelas (keputusan) yaitu angin, cuaca, dam kelembapan. Kemudian dilakukan perhitungan *pairwise SU* untuk masing-masing fitur yang relevan guna menyeleksi fitur-fitur yang redundan.

Tabel 2.2 Ilustrasi Nilai *Pairwise SU*

	Cuaca	Kelembapan	Angin
Cuaca	1	0,0161	0,0045
Kelembapan	0,0161	1	0
Angin	0,0045	0	1

a. Pasangan fitur cuaca dan kelembapan:

Nilai *SU*(cuaca,kelembapan) < *SU*(keputusan,kelembapan),maka fitur kelembapan bukan merupakan fitur redundan.

b. Pasangan fitur cuaca dan angin:

Nilai *SU*(cuaca,angin) < *SU*(keputusan,angin), maka fitur kelembapan bukan merupakan fitur redundan.

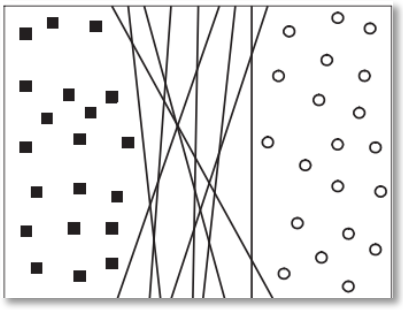
c. Pasangan fitur kelembapan dan angin:

Nilai *SU*(kelembapan,angin) < *SU*(keputusan,angin), maka fitur angin bukan merupakan fitur redundan.

Berdasarkan hasil ilustrasi diatas, diperoleh fitur-fitur yang relevan terhadap kelas dengan nilai *threshold* 0,05 yaitu cuaca, kelembapan, dan angin. Dari ketiga fitur yang relevan tidak ditemukan adanya fitur yang redundan sehingga hanya fitur temperatur yang dibuang dari hasil seleksi fitur FCBF.

2.2 *Support Vector Machines* (SVM)

Support Vector Machines (SVM) adalah suatu algoritma *machine learning* yang diperkenalkan pada tahun 1992 untuk masalah klasifikasi (Vapnik dkk, 1995). Algoritma SVM menggunakan *mapping non* linier untuk mentransformasi data training asal ke dimensi yang lebih besar, dimana dalam dimensi baru tersebut, SVM melakukan pencarian pemisah *hyperplane* optimum yang linier. *Hyperplane* merupakan garis batas pemisah data antar kelas. Dengan sebuah dimensi yang lebih tinggi dari dimensi asalnya, data dari dua kelas yang berbeda selalu dapat dipisahkan oleh sebuah *hyperplane* (Han dkk, 2012).

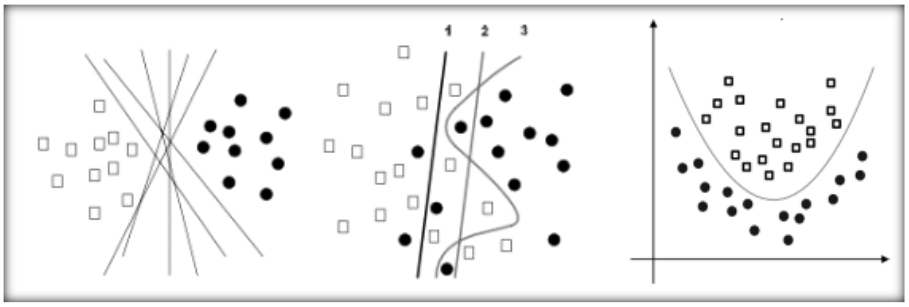


Gambar 2.2 Ilustrasi *Hyperplane* yang Mungkin pada Suatu Pengklasifikasian (Tan dkk, 2006)

Berdasarkan gambar 2.2, terlihat bahwa terdapat tak hingga kemungkinan *hyperplane* yang bisa dipilih. Walaupun pada saat melakukan *training* data menghasilkan *error* sama dengan nol, tidak ada garansi bahwa *hyperplane* yang dipilih tersebut akan menunjukan hasil yang sama baiknya saat memisahkan data baru (Tan dkk, 2006).

SVM menemukan *hyperplane* ini menggunakan sekumpulan *support vector* dan sebuah *margin* yang didefinisikan oleh sekumpulan *support vector* tersebut. *Margin* didefinisikan sebagai jarak antara *hyperplane* dengan data terdekat pada masing-masing kelas. Bidang pembatas pertama membatasi kelas pertama dan bidang pembatas kedua membatasi kelas kedua sedangkan data yang berada pada bidang pembatas merupakan vektor-vektor yang terdekat dengan *hyperplane* terbaik disebut dengan *Support Vector* (Han dkk, 2012). *Hyperplane* terbaik diperoleh dengan cara memaksimalkan *margin* (mencari *margin* yang paling luas), karena memaksimalkan *margin* antara dua set obyek akan meningkatkan probabilitas pengelompokan secara benar dari data *testing* (Santosa, 2007).

Algoitma SVM dapat bekerja pada kasus klasifikasi linier maupun *nonlinier*. Pada klasifikasi linier, menurut Tan dkk (2006) SVM dapat dibedakan menjadi dua yaitu *linearly separable* dan *linearly non separable*. Gambar 2.3 menunjukan ilustrasi dari klasifikasi linier dan *non* linier



Gambar 2.3 Klasifikasi SVM: (kiri) Klasifikasi *Linearly Separable*; (tengah) *Linearly Non Separable*; (kanan) *Non* Linier (Hardle dkk, 2014)

2.2.1 SVM *Linearly Separable*

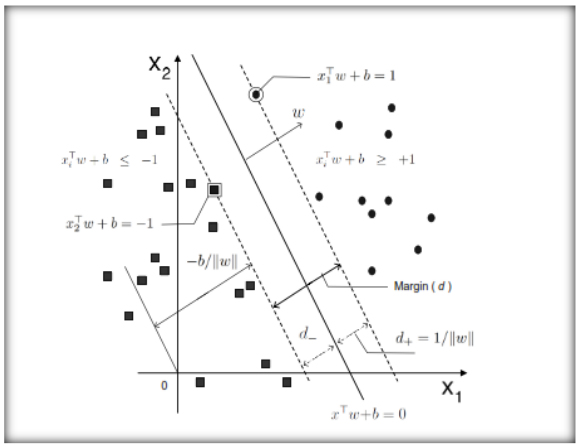
Misalkan setiap pengamatan terdiri atas sepasang p prediktor $\mathbf{x}_i = \left(x_{i1}, x_{i2}, ..., x_{ip}\right) \in \mathrm{R}^p$, $i = 1,2,...,n$ dan dihubungkan dengan kelas label $y_i \in \mathbf{y} = \{-1,1\}$, maka dapat dinyatakan dalam himpunan berikut:

$$D_n = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\} \in X \times \{-1,1\}$$

Adalah sebuah *dataset* yang akan digunakan untuk memprediksi kelas label **y** untuk setiap pengamatan (Hardle dkk, 2014). Konsep utama dalam mendefinisikan *classifier* yang bersifat linier yaitu *dot product*, yang juga biasa disebut *inner product* atau *scalar product* diantara dua vektor didefinisikan sebagai $\mathbf{x}^T\mathbf{w} = \sum_i x_i w_i$. Fungsi pemisah (*hyperplane*) dari sebuah klasifikasi linier dapat dituliskan dalam bentuk berikut:

$$f(x) = \mathbf{x}^T\mathbf{w} + b = 0 \qquad (2.6)$$

Dimana **w** diketahui sebagai bobot vektor dan b disebut bias.



Gambar 2.4 *Hyperplane* Pemisah $\mathbf{x}^T\mathbf{w} + b = 0$ dan *Margin* Dalam Kasus *Linearly Separable* (Haerdle, W.K., dkk, 2014)

Bentuk fungsi pada persamaan (2.6) adalah sebuah garis dalam dua dimensi, sebuah bidang (*plane*) pada tiga dimensi, dan lebih generalnya adalah sebuah *hyperplane* dalam dimensi yang lebih tinggi. Persamaan (2.6) membagi ruang ke dalam 2 region sebagaimana terlihat pada gambar 2.4. hyperplane dikatakan linier jika merupakan fungsi linier dalam input x_i . Guna menentukan *support vector*, maka dipilih **w**,*b* yang dapat membuat *margin* menjadi maksimal, dimana margin sama dengan $d_- + d_+$. Tanda (-) dan (+) menyatakan kedua daerah yang dipisahkan *hyperplane* (Hardle dkk, 2014).

Fase *training* dari SVM termasuk pengestimasian parameter *b* dan **w** dari data yang dilatih. Parameter harus dipilih sebagaimana mungkin sehingga dua kondisi berikut terpenuhi.

$$\begin{aligned} \mathbf{x}^T\mathbf{w} + b \geq 1 \quad &\text{jika } y_i = 1, \\ \mathbf{x}^T\mathbf{w} + b \leq -1 \ &\text{jika } y_i = -1 \end{aligned} \tag{2.7}$$

Kondisi diatas mengharuskan bahwa semua pengamatan yang dilatih dari kelas *y=1* atau lingkaran pada gambar 2.4 harus berlokasi di atas *hyperplane*, sementara kelas *y=-1* atau persegi pada gambar 2.4 harus berlokasi di bawah *hyperplane*. Kedua fungsi kendala diatas dapat digabungkan menjadi bentuk berikut

$$y_i(\mathbf{x}^T\mathbf{w} + b) \geq 1, \quad i = 1,2,...,n \tag{2.8}$$

Dimana:

$$\mathbf{x}_i^{\,T} = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{ip} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

Pada gambar 2.4, $\|\mathbf{w}\|$ merupakan panjang vektor dari **w** atau dapat didefinisikan $\|\mathbf{w}\| = \sqrt{\mathbf{w}^T\mathbf{w}} = \sqrt{w_1^{\,2} + w_2^{\,2} + ... + w_p^{\,2}}$. Nilai dari *margin* (*d*) dapat dihitung dengan mengurangkan jarak tegak lurus bidang pembatas kelas positif dari titik asal dengan jarak tegak lurus bidang pembatas kelas negatif dari titik asal. Nilai *margin* pada gambar 2.3 yaitu

$$d = \frac{(1-b)-(-1-b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{2.9}$$

Sehingga bidang pemisah (*separating hyperplane*) optimum diperoleh dengan memaksimalkan persamaan (2.9), atau secara ekuivalen meminimalkan fungsi obyektif berikut

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} \tag{2.10}$$

dengan fungsi kendala pada persamaan (2.8). Sejak diketahui bahwa fungsi obyektif pada persamaan (2.10) berbentuk kuadratik dan fungsi kendala pada persamaan (2.8) berbentuk linier maka masalah ini dikenali sebagai masalah optimasi *convex*, yang mana dapat diselesaikan dengan menggunakan metode *Langrangian* standar. Fungsi obyektif baru yang diketahui sebagai masalah optimasi *Langrangian* yaitu

$$L_P(\mathbf{w},b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} a_i \left(y_i(\mathbf{x}_i^{\,T}\mathbf{w} + b) - 1 \right) \tag{2.11}$$

Dimana parameter a_i disebut sebagai pengali *Langrange*. Untuk meminimalkan *Langrangian*, maka akan dicari turunan pertama dari fungsi L_p terhadap **w** dan *b*.

$$\frac{\partial L_P(\mathbf{w},b)}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} a_i y_i \mathbf{x}_i \tag{2.12}$$

$$\frac{\partial L_P(\mathbf{w},b)}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} a_i y_i = 0 \tag{2.13}$$

Karena pengali *Langrange* a_i tidak diketahui, maka **w** dan *b* masih belum dapat dicari solusinya. Untuk itu formula *Langrange* L_p (masalah *primal*) akan diubah ke dalam L_D (masalah *dual*) dengam mensubtitusi persamaan (2.12) dan (2.13) ke persamaan (2.11), sehingga diperoleh *Langrangian* untuk masalah *dual* yaitu

$$L_D(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j \qquad (2.14)$$

Komponen kedua pada persamaan (2.14) diatas memiliki tanda negatif yang berarti bahwa masalah minimasi pada fungsi L_p akan diubah ke masalah maksimasi L_D. Jadi persoalan pencarian bidang pemisah (*hyperplane*) terbaik dapat dirumuskan pada persamaan berikut

$$\max_a \; L_D(a) = \max\left(\sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j \right) \qquad (2.15)$$

Dengan fungsi kendala

$$a_i \geq 0, \quad \sum_{i=1}^{n} a_i y_i = 0 \qquad (2.16)$$

Masalah optimasi diatas dapat diselesaikan menggunakan teknik numerik seperti *quadratic programming* (Tan dkk, 2006). Meminimumkan fungsi *Langrangian primal* L_p dan memaksimalkan fungsi *Langrangian dual* L_D akan memberikan solusi yang sama ketika masalah optimasi adalah *convex*.

Nilai a_i yang diperoleh, nantinya akan digunakan untuk mencari nilai **w** menggunakan persamaan berikut.

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{a}_i y_i \mathbf{x}_i \qquad (2.17)$$

Suatu pengamatan baru dapat diklasifikasikan menggunakan fungsi klasifikasi berikut

$$\hat{f}(\mathbf{x}_i) = sign\,(\mathbf{x}_i^T \hat{\mathbf{w}} + \hat{b}) \qquad (2.18)$$

dimana

$$\hat{b} = \frac{1}{N_{sv}}\left(\sum_{i=1}^{N_{sv}} \frac{1}{y_i} - (\mathbf{x}_i{}^T \mathbf{w}) \right) \qquad (2.19)$$

2.2.2 SVM *Linearly Non Separable*

Dalam kasus pada gambar 2.3 (tengah), formulasi yang disajikan pada kasus *linearly separable* perlu dimodifikasi untuk membuat fungsi pemisah yang dapat memberikan toleransi pada *training error* yang kecil , menggunakan metode yang diketahui sebagai pendekatan *soft margin*. Metode ini mengijinkan SVM untuk membangun sebuah fungsi pemisah yang linier bahkan dalam situasi dimana kelas-kelas tidak dapat dipisahkan secara linier. Untuk melakukan ini algoritma learning SVM harus mempertimbangkan *trade-off* diantara lebarnya *margin* dan besarnya *training error* dalam pembentukan fungsi pemisah yang linier (Tan dkk, 2006).

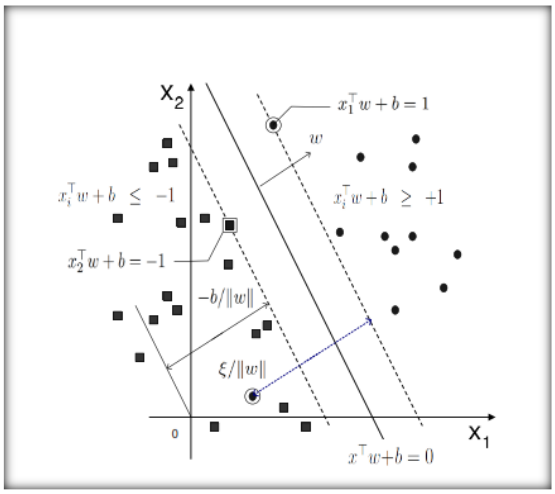
Hardle, Prastyo dan Hafner (2014) menyatakan pada kasus *linearly non separable*, fungsi kendala pada persamaan (2.7) harus dimodifikasi dengan menambahkan variabel *slack* ξ_i yang mempresentasikan penalti dari ketelitian pemisahan yang mana mengijinkan sebuah titik untuk berada dalam sebuah *margin error* ($0 \leq \xi_i \leq 1$) atau salah diklasifikasikan $\xi_i > 1$. Pada kasus ini fungsi kendala akan berubah menjadi

$$\begin{aligned} \mathbf{x}^T\mathbf{w} + b &\geq 1 - \xi_i \quad \text{jika } y_i = 1, \\ \mathbf{x}^T\mathbf{w} + b &\leq -1 + \xi_i \text{ jika } y_i = -1 \end{aligned} \tag{2.20}$$

Dimana $\xi_i \geq 0$. Kemudian kedua fungsi kendala diatas dapat digabungkan menjadi

$$\begin{aligned} y_i(\mathbf{x}^T\mathbf{w} + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned} \tag{2.21}$$

Penalti untuk kesalahan klasifikasi berhubungan dengan jarak dari suatu titik yang salah diklasifikasikan dari suatu *hyperplane* kanonik yang membatasi kelasnya.



Gambar 2.5 *Hyperplane* Pemisah $\mathbf{x}^T\mathbf{w} + b = 0$ dan *Margin* Dalam Kasus *Linearly Non Separable* (Hardle dkk, 2014).

Fungsi obyektif yang berkorespondensi dengan pemaksimalan *margin* dengan sebuah penalti diformulasikan sebagai berikut

$$\min_{\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \qquad (2.22)$$

dengan fungsi kendala yang ada pada persamaan (2.21). Variabel *slack* tak negatif ξ_i mengijinkan suatu titik untuk berada di sisi yang salah dari *soft margin* $(\mathbf{x}_i^T\mathbf{w} + b = \pm 1)$ dan parameter C menyajikan suatu penalti terhadap suatu pengklasifikasian yang salah pada pengamatan yang dilatih. Peminimalisasian fungsi tujuan pada persamaan (2.22) berdasarkan fungsi kendala pada persamaan (2.21) akan menyediakan kemungkinan *margin* terbesar dalam kasus dimana suatu kesalahan klasifikasi diijinkan. Parameter C dipilih berdasarkan performansi model SVM pada suatu proses validasi data *training*. Fungsi *Langrangian* untuk masalah *primal* yaitu

$$L_P(\mathbf{w},b,\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}a_i\left(y_i(\mathbf{x}^T\mathbf{w} + b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\mu_i\xi_i \qquad (2.23)$$

dimana komponen pertama dan kedua pada persamaan yaitu fungsi tujuan yang akan diminimalkan, komponen ketiga yaitu pertaksamaan pada fungsi kendala yang berasosiasi dengan variabel *slack*, dan komponen ke-empat yaitu syarat tak

negatif dari nilai ξ_i. a_i dan μ_i adalah pengali *Langrange*. Fungsi kendala pertaksamaan dapat ditransformasi menjadi bentuk fungsi kendala berupa persamaan menggunakan kondisi Karush KuhnTucker (KKT) berikut

$$\begin{aligned} \xi_i \geq 0, \quad a_i \geq 0, \quad \mu_i \geq 0 \\ a_i\left(y_i(\mathbf{x}^T\mathbf{w}+b)-1+\xi_i\right)=0 \\ \mu_i\xi_i = 0 \end{aligned} \tag{2.24}$$

Turunan pertama dari persamaan (2.23) terhadap **w**, *b*, dan ξ_i yang disama-dengankan nol akan menghasilkan persamaan berikut

$$\frac{\partial L_P(\mathbf{w},b,\xi)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} a_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \mathbf{w} = \sum_{i=1}^{n} a_i y_i \mathbf{x}_i \tag{2.25}$$

$$\frac{\partial L_P(\mathbf{w},b,\xi)}{\partial b} = -\sum_{i=1}^{n} a_i y_i = 0 \quad \Rightarrow \sum_{i=1}^{n} a_i y_i = 0 \tag{2.26}$$

$$\frac{\partial L_P(\mathbf{w},b,\xi)}{\partial \xi_i} = C - a_i - \mu_i = 0 \quad \Rightarrow \mu_i = C - a_i \tag{2.27}$$

Masalah *primal* ditransformasi ke dalam masalah *dual* dengan mensubtitusikan persamaan (2.25), (2.26), (2.27) ke fungsi *Langrangian primal* pada persamaan (2.18) sebagai berikut

$$\begin{aligned} L_D(a) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j - \sum_{i=1}^{n} a_i y_i \mathbf{x}_i{}^T \sum_{j=1}^{n} a_j y_j \mathbf{x}_j + C\sum_{i=1}^{n}\xi_i \\ + \sum_{i=1}^{n} a_i - \sum_{i=1}^{n} a_i \xi_i + \sum_{i=1}^{n} \mu_i \xi_i \\ = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j + \sum_{i=1}^{n} \xi_i \left(C - a_i - \mu_i\right) \end{aligned} \tag{2.28}$$

Karena komponen terakhir pada persamaan (2.28) sama dengan nol, maka

$$L_D(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j \tag{2.29}$$

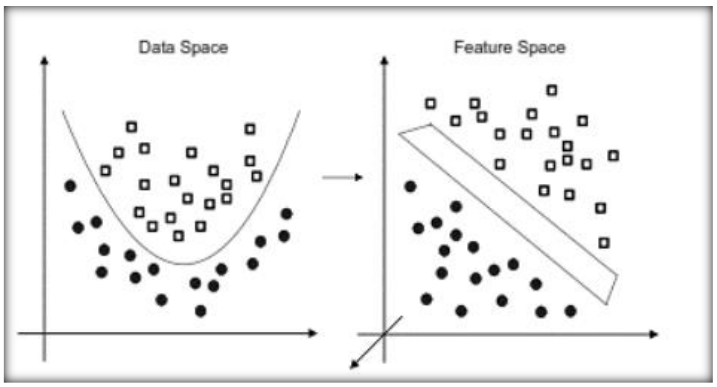
yang mana menjadi masalah *dual* yang sama dengan pada kasus *linearly separable* hanya saja fungsi kendala nya berbeda. Fungsi kendala pada masalah *dual* yaitu

$$0 \leq a_i \leq C, \quad \sum_{i=1}^{n} a_i y_i = 0 \tag{2.30}$$

Permasalahan *dual* di atas kemudian dapat diselesaikan secara numerik menggunakan *quadratic programming* untuk mencari nilai a_i. Pengamatan $\mathbf{x}_i$ untuk $a_i > 0$ yaitu suatu titik (*support vector*) yang berada di atas atau di dalam margin ketika *soft margin* digunakan (Scholkopf dan Smola, 2002).

2.2.3 SVM *Non Linear*

Menurut Hardle, Prastyo dan Hafner (2014), SVM juga bisa digeneralisasikan ke kasus nonlinier. Seperti pada gambar 2.3 (kanan), dalam kasus dimana sebuah pemisah linier tidak sesuai (tidak bisa diterapkan), SVM bisa mentransformasikan vektor input, **x**, ke sebuah ruang fitur berdimensi tinggi. Sebuah transformasi nonlinier, Φ dibutuhkan untuk memetakan data dari ruang fitur asalnya ke ruang baru berdimensi yang lebih tinggi. Dalam dimensi tersebut, kemudian SVM membangun sebuah batas keputusan linier yang memisahkan pengamatan ke masing-masing kelasnya (Tan dkk, 2006).



Gambar 2.6 Pemetaan Ruang Fitur Data Input Dua Dimensi (kiri) ke Ruang Fitur Tiga Dimensi (kanan) $\mathbf{R}^2 \rightarrow \mathbf{R}^3$ (Hardle dkk, 2014)

Klasifikasi nonlinier pada gambar (2.6), adalah hasil suatu pemetaan data dengan struktur nonlinier melalui suatu fungsi $\Phi : \mathbf{R}^p \rightarrow \mathbf{H}$, dimana $\mathbf{H}$ merupakan suatu ruang berdimensi tinggi. Perhatikan bahwa semua *vector training* $\mathbf{x_i}$ yang terdapat pada persamaan (2.29) sebagai *scalar product* dari bentuk $\mathbf{x}_i^T \mathbf{x}_j$. Pada SVM nonlinier, *scalar product* tersebut ditransformasikan ke bentuk $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$.

Proses transformasi tersebut disebut sebagai *"Kernel Trick"* (Scholkopf dan Smola, 2002). Proyeksi $\Phi : \mathbf{R}^p \to \mathbf{H}$, memastikan bahwa *scalar product* $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ bisa di sajikan oleh fungsi kernel.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \tag{2.31}$$

Jika suatu fungsi kernel *k* pada persamaan (2.31), dapat digunakan tanpa perlu mengetahui fungsi transformasi Φ secara eksplisit.

Diberikan suatu kernel *k* dan suatu data set $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in X$, maka matriks $K = \left(k\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)_{ij}$ yang berukuran *n* x *n* disebut sebagai matriks kernel dari *k* untuk suatu data $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$. Untuk menghasilkan fungsi klasifikasi nonlinier dalam data *space*, sebuah bentuk yang lebih umum dihasilkan dengan menerapkan *Kernel trick* ke persamaan (2.29) sebagai berikut

$$\max_a L_D = \max_a \left(\sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j k\left(\mathbf{x}_i, \mathbf{x}_j\right) \right) \tag{2.32}$$

Dengan fungsi kendala

$$\begin{aligned} &0 \le a_i \le C \quad , i = 1, ..., n \\ &\sum_{i=1}^{n} a_i y_i = 0 \end{aligned} \tag{2.33}$$

Fungsi klasifikasi pada SVM non linier yakni

$$f(\mathbf{x}) = \sum_{i=1}^{n} a_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b \tag{2.34}$$

dimana fungsi Kernel yang biasa digunakan yaitu

1. Kernel Linier

   $$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \tag{2.35}$$

2. Kernel *Polynomial*

   $k(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x} + r)^p$, $\gamma > 0$, *p* adalah derajad polynomial.

3. Kernel *Radial Basis Function* (RBF)

   $$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2), \quad \gamma > 0 \tag{2.36}$$

4. Kernel *Sigmoid*

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x} + r), \quad \gamma > 0$$

Dimana γ, r, d adalah parameter Kernel.

Pemilihan fungsi Kernel yang tepat merupakan hal yang sangat penting karena akan menentukan ruang fitur dimana fungsi *classifier* akan dicari. Sepanjang fungsi kernelnya sesuai, SVM akan beroperasi secara benar meskipun tidak tahu pemetaan yang digunakan (Santosa, 2007). Menurut Hsu, Chang dan Lin (2003), fungsi Kernel yang direkomendasikan untuk diuji pertama kali adalah fungsi Kernel RBF karena dapat memetakan hubungan tidak linier, RBF lebih robust terhadap outlier karena fungsi Kernel RBF berada antara selang $(-\infty, \infty)$ sedangkan fungsi kernel yang lain memiliki rentang antara (-1 sampai dengan 1). Gaussian RBF juga efektif menghindari *overfitting* dengan memilih nilai yang tepat untuk parameter *C* dan γ dan RBF baik digunakan ketika tidak ada pengetahuan terdahulu.

2.3 *Synthetic Minority Oversampling Technique (*SMOTE)

SMOTE merupakan salah satu metode dalam penanggulangan data *imbalance* yang diusulkan oleh Chawla (2002). Ide dasar dari SMOTE yaitu menambah jumlah sampel pada kelas minor agar setara dengan kelas mayor dengan cara membangkitkan data *synthetic* berdasarkan tetangga terdekat *k-nearest neighbour* dimana tetangga terdekat dipilih berdasarkan jarak *euclidean* antara kedua data (Chawla dkk, 2003).

Misalkan diberikan data dengan *p* variabel yaitu $\mathbf{x}^T = [\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_p]$ dan $\mathbf{z}^T = [\mathrm{z}_1, \mathrm{z}_2, ..., \mathrm{z}_p]$ maka jarak *euclidean d*(x,z) secara umum sebagai berikut:

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{(\mathrm{x}_1 - \mathrm{z}_1)^2 + (\mathrm{x}_2 - \mathrm{z}_2)^2 + ... + (\mathrm{x}_p - \mathrm{z}_p)^2}$$

Pembangkitan data *synthetic* dilakukan dengan menggunakan persamaan berikut:

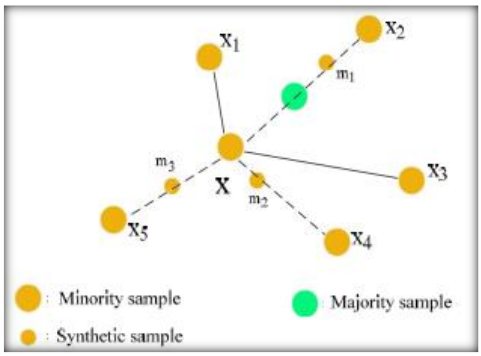
$$\mathbf{x}_{syn} = \mathbf{x}_i + (\mathbf{x}_{knn} - \mathbf{x}_i)\gamma \qquad (2.37)$$

Dimana: $\mathbf{x}_{syn}$ adalah data *synthetic*

$\mathbf{x}_i$ adalah data ke-*i* dari kelas minor

$\mathbf{x}_{knn}$ adalah data dari kelas minor yang memiliki jarak terdekat dari $\mathbf{x}_i$

γ adalah bilangan random antara 0 dan 1



Gambar 2.7 Ilustrasi SMOTE (Cui dkk, 2014)

Pada gambar 2.7, X menyatakan sebuah sampel dan $X_1,...,X_5$ adalah tetangga terdekat dari sampel X. SMOTE membangkitkan data baru (data *synthetic*) m_i melalui sebuah garis diantara X dan masing-masing tetangga terdekatnya.

2.4 *Boosting*

Boosting merupakan salah satu metode *ensemble* yang digunakan untuk mengimprovisasi performa suatu algoritma *learning* dengan mengkombinasikan kumpulan *classifier* lemah guna membentuk suatu *classifier* akhir yang kuat. *Boosting* mengasumsikan ketersediaan dari suatu algoritma *learning* yang lemah (suatu algoritma *learning* yang menghasilkan model *classifier* lemah atau model yang kurang akurat dalam memprediksi suatu *training*). Ide utama didalam proses *boosting* yaitu memilih sekumpulan data *training* (sampel *training*) dengan beberapa cara untuk kemudian dipelajari oleh suatu *base learner*, dimana *base learner* tersebut dipaksa menarik sesuatu yang baru tentang sampel tersebut setiap kali *base learner* itu dipanggil. Proses ini dapat dicapai dengan memilih sampel *training* yang diharapkan dapat membuat performa dari *base classifier* menjadi sangat buruk bahkan lebih buruk dari pada performa *base classifier* secara reguler. Jika hal ini dapat dicapai, kemudian pada iterasi selanjutnya diharapkan *base learner* dapat menghasilkan suatu *base classifier* baru yang secara signifikan berbeda dari pendahulunya. Hal ini dikarenakan meskipun *base learner* diharapkan sebagai sesuatu algoritma *learning* yang lemah dan medioker, namun

base learner ini diharapkan pula dapat memberikan *output* suatu *classifier* yang membuat prediksi *nontivial* (Schapire dan Freund, 2012).

AdaBoost adalah salah satu representasi algoritma *boosting*. Algoritma *AdaBoost* diperkenalkan oleh Freund dan Schapire (1995), memecahkan banyak masalah *practical* dari algoritma *boosting* terdahulu. Algoritma ini mengambil *input* sekumpulan data *training* $(\mathbf{x}_1, y_1),...,(\mathbf{x}_m, y_m)$, dimana setiap $\mathbf{x}_i \in \mathbf{X}$ dan $y_i \in \mathbf{y} = \{-1,+1\}$.Adaboost memanggil sebuah *base learner* secara berulang dalam suatu urutan iterasi t=1,..T. Dalam memilih sampel *training* yang akan digunakan oleh *base learner* pada setiap iterasi *boosting* berdasarkan data *training* yang disediakan, *AdaBoost* menggunakan sebuah distribusi dari sampel *training*. Distribusi yang digunakan pada iterasi ke-*t* dinyatakan dalam D_t, dan bobot untuk sampel *training* ke-*i* dinyatakan dalam $D_t(i)$. Secara intuitif, bobot ini adalah suatu ukuran dari pentingnya sampel ke-*i* terklasifikasikan secara benar pada suatu iterasi. Pada inisialisasi, semua bobot diatur sama besarnya, kemudian pada setiap iterasi bobot-bobot dari sampel yang terklasifikasikan salah akan dinaikan, dengan kata lain sampel yang sulit diklasifikasikan akan memperoleh bobot yang lebih besar sehingga memaksa *base learner* untuk memberikan perhatian lebih terhadap sampel-sampel tersebut. Tugas dari *base learner* yaitu menemukan sebuah *weak hypothesis* $h_t : X \to \{-1,+1\}$ sesuai distribusi D_t. Kebaikan dari suatu *weak hypothesis* diukur berdasarkan *error*

$$e_t = \Pr_{i \sim D_t}\left[h_t(\mathbf{x}_i) \neq y_i\right] = \sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_t(i) \qquad (2.38)$$

Perhatikan bahwa *error* diukur dengan mempertimbangkan distribusi D_t dari hasil *training* suatu *weak classifier*. Dalam praktiknya, beberapa *weak learner* mungkin merupakan suatu algoritma yang bisa menggunakan bobot-bobot D_t pada sampel *training*. Sebagai alternatif, saat hal ini tidak memungkinkan, suatu *subset* dari sampel *training* bisa diambil sampel berdasarkan D_t, dan sampel-sampel ini bisa digunakan untuk melatih *weak lerner*. Berikut algoritma *Boosting AdaBoost* secara umum:

Input: $(\mathbf{x}_i, y_i),...,(\mathbf{x}_m, y_m)$ dimana $\mathbf{x}_i \in \mathbf{X}$ dan $y_i \in \mathbf{y} = \{-1,+1\}$

Inisialisasi $D_1(i) = 1/m$

Untuk t = 1,...., T

1. Latih *weak learner* menggunakan distribusi D_t
2. Dapatkan *weak hypothesis* $h_t : X \to \{-1,+1\}$ dengan *error* pada persamaan (2.38). Jika diperoleh nilai $e_t > 0{,}5$, maka proses *learning* berhenti.
3. Hitung $a_t = \frac{1}{2}\ln\left(\frac{1-e_t}{e_t}\right)$ (2.39)
4. *Update* nilai bobot

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-a_t} & jika\ h_t(\mathbf{x}_i) = y_i \\ e^{a_t} & jika\ h_t(\mathbf{x}_i) \neq y_i \end{cases}$$

$$D_{t+1}(i) = \frac{D_t(i)\exp(-a_t y_i h_t(\mathbf{x}_i))}{Z_t} \quad (2.40)$$

Dimana Z_t adalah sebuah konstanta normalisasi yang membuat $\sum_{i=1}^{m} D_{t+1}(i) = 1$

Output dari hipotesis akhir yaitu

$$H(\mathbf{x}) = sign\left(\sum_{t=1}^{T} a_t h_t(\mathbf{x})\right) \quad (2.41)$$

Gambar 2.8 Algoritma *AdaBoost* (Schapire dan Freund, 1995)

AdaBoost secara linier mengkombinasikan semua *weak classifier* menjadi sebuah hipotesis akhir $H(\mathbf{x})$. Hipotesis akhir $H(\mathbf{x})$ adalah sebuah *majority vote* yang terboboti dari *T weak hypothesis* dimana a_t adalah suatu bobot yang ditugaskan untuk h_t. Nilai $a_t \geq 0$ jika $e_t \leq 1/2$ dan a_t akan lebih besar jika e_t lebih kecil (Freund dan Schapire, 1999).

Bagaimanapun algoritma *boosting* seperti *AdaBoost* dapat menghasilkan *overfitting* dan masalah generalisasi karena algoritma tersebut mencoba untuk memaksimalkan akurasi aritmatik. *Error* dari *classifier* e_t dan performansi dari *classifier* a_t diukur berdasarkan rata-rata aritmatik. Suatu ukuran berdasarkan rata-rata aritmatik mungkin menjadi tidak valid sebagai suatu fungsi obyektif yang berguna karena fungsi obyektif yang diukur berdasarkan ukuran aritmatik akan membangkitkan suatu fungsi klasifikasi yang bias terhadap kelas mayor atau kelas dengan *similarity* yang tinggi antar sampel-sampelnya. Khususnya saat suatu algoritma *boosting* diaplikasikan setelah SMOTE dilakukan. Gagasan akurasi geometrik bisa dipertimbangkan untuk meringankan masalah ini (Kim dkk, 2015)

2.5 *SMOTEBoost*

Algoritma ini diusulkan oleh Chawla pada tahun 2003. Pada tiap iterasinya, SMOTEBoost memperkerjakan algoritma SMOTE guna menambah probabilitas dari terpilihnya sampel-sampel yang sulit diklasifikasikan yang mana berasal dari kelas minoritas (positif) kedalam *training set* dimana pada iterasi dalam AdaBoost, *training set* tersebut kebanyakan didominasi oleh sampel dari kelas mayoritas (negatif) yang sulit diklasifikasikan.

Tujuan dari penggabungan algoritma SMOTE dan Adaboost yaitu untuk meningkatkan nilai True Positive (TP) rate (Chawla, 2003). SMOTEBoost sukses mengkombinasikan AdaBoost dan SMOTE. Sementara Adaboost mencoba meningkatkan akurasi dari *classifier* dengan fokus pada pengamatan yang sulit diklasifikasikan yang berasal dari kedua kelas, SMOTE mencoba meningkatkan performansi dari *classifier* hanya pada pengamatan pada kelas minoritas. Oleh karena itu dalam beberapa iterasi *boosting* berurutan, SMOTEBoost mampu membuat daerah keputusan yang lebih luas untuk kelas minoritas dibandingkan metode *boosting* standar. SMOTEBoost pada awalnya menggunakan prosedur iterasi dari dari *boosting* AdaBoost.M2 oleh (Freund dan Schapire, 1996). Dalam prosedur iterasi AdaBoost.M2, hasil klasifikasi dari komponen *classifier* harus terlebih dahulu dibawa ke dalam bentuk probabilitas [0,1] guna nantinya dipakai dalam menghitung *pseudo loss*. Hal ini berbeda dengan pendekatan AdaBoost

original. Perubahan ini dimaksutkan guna mengatasi asumsi dalam AdaBoost dimana komponen *classifier* harus memiliki akurasi yang lebih baik dari random *guessing* (error < 0.5) yang mana hal ini akan sering ditemui pada kasus klasifikasi multikelas (Freund dan Schapire, 1996). Sehingga, pada *boosting* SMOTEBoost bisa saja dilakukan modifikasi dengan menggunakan prosedur iterasi AdaBoost original guna memperoleh prosedur iterasi yang lebih simpel. Berikut algoritma SMOTEBoost dengan modifikasi menggunakan prosedur iterasi AdaBoost original oleh (Schapire dan Freund, 1995) ditampilkan pada Gambar 2.8

Input: $(\mathbf{x}_i, y_i),...,(\mathbf{x}_m, y_m)$ dimana $\mathbf{x}_i \in \mathbf{X}$ dan $y_i \in \mathbf{y} = \{-1,+1\}$

Inisialisasi $D_1(i) = 1/m$

Untuk t = 1,...., T

1. Memodifikasi distribusi D_t dengan mereplikasi pengamatan kelas minor menggunakan algoritma SMOTE.
2. Latih *weak learner* menggunakan distribusi D_t
3. Dapatkan *weak hypothesis* $h_t : X \to \{-1,+1\}$ dengan *error* pada persamaan.

$$e_t = \Pr_{i \sim D_t}\left[h_t(\mathbf{x}_i) \neq y_i\right] = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i) \qquad (2.42)$$

Jika diperoleh nilai $e_t > 0{,}5$, maka proses *learning* berhenti.

4. Hitung $a_t = \frac{1}{2}\ln\left(\frac{1-e_t}{e_t}\right)$ (2.43)
5. *Update* nilai bobot

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-a_t} & jika\ h_t(\mathbf{x}_i) = y_i \\ e^{a_t} & jika\ h_t(\mathbf{x}_i) \neq y_i \end{cases}$$

$$D_{t+1}(i) = \frac{D_t(i)\exp(-a_t y_i h_t(\mathbf{x}_i))}{Z_t} \qquad (2.44)$$

Dimana Z_t adalah sebuah konstanta normalisasi yang membuat $\sum_{i=1}^{m} D_{t+1}(i) = 1$

Output dari hipotesis akhir yaitu

$$H(\mathbf{x}) = sign\left(\sum_{t=1}^{T} a_t h_t(\mathbf{x})\right) \qquad (2.45)$$

Gambar 2.9 Algoritma SMOTEBoost dengan prosedur iterasi Adaboost original (Schapire dan Freund, 1995)

2.6 *K-Fold Cross Validation*

Cross-validation adalah metode statistik untuk mengevaluasi dan membandingkan algoritma *learning* dengan membagi data menjadi dua bagian

yaitu data *training* yang digunakan untuk pelatihan dan data *testing* yang digunakan untuk memvalidasi model. Bentuk dasar *cross-validation* adalah *k-fold cross-validation*. Pada k-*fold cross-validation*, data dipartisi secara acak menjadi k bagian yang bersifat *mutually exlclusive* $D_1, D_2, ..., D_k$, yang masing masing memiliki ukuran yang sama. Proses *training* dan *testing* dilakukan sebanyak *k* kali. Dalam iterasi ke-*i*, data partisi D_i diposisikan sebagai data *testing*, sementara partisi lain yang tersisa secara kolektif digunakan untuk melatih model. Misalkan pada iterasi pertama, D_1 digunakan sebagai data *testing*, sementara $D_2, D_3, ..., D_k$ digabungkan untuk digunakan sebagai data *training* untuk menghasilkan model, yang kemudian diuji pada data D_1. Pada iterasi kedua, data $D_1, D_3, ..., D_k$ digabungkan untuk digunakan sebagai data *training* dan kemudian model yang dihasilkan dilakukan pengujian menggunakan data D_2. Akurasi klasifikasi model diperoleh dengan cara merata-ratakan akurasi dari setiap iterasi (Han dkk, 2012).

2.7 Evaluasi Performansi

Akurasi adalah suatu matrik evaluasi yang sangat penting untuk menaksir performansi suatu hasil klasifikasi. Lokanayaki dan Malathi (2014) mengatakan bahwa evaluasi performansi suatu *classifier* pada kelas *imbalance* dapat diukur menggunakan *G-mean*. Ukuran ukuran performansi tersebut dapat dihitung berdasarkan suatu matriks konfusi (*confusion matrix*).

Tabel 2.3 Matriks Konfusi

	Predicted Positive	*Predicted Negative*
Actual Positive	TP	FN
Actual Negative	FP	TN

Keterangan:

TP : *True Positive* (jumlah prediksi benar pada kelas positif)

TN : *True Negative* (jumlah prediksi benar pada kelas negatif)

FP : *False Positive* (jumlah prediksi salah pada kelas positif)

FN : *False Negative* (jumlah prediksi salah pada kelas negatif)

Berdasarkan tabel 2.1, ukuran performansi akurasi *overall*, *error rate*, *sensitivity*, dan *specificity* dapat dihitung.

$$\text{akurasi } overall = \frac{TN + TP}{TN + TP + FN + FP} \tag{2.46}$$

$$error\ rate = \frac{FP + FN}{P + N} \tag{2.47}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{2.48}$$

$$specificity = \frac{TN}{TN + TP} \tag{2.49}$$

Sensitivity merupakan ukuran performansi kelas positif, sedangkan *specificity* merupakan ukuran performansi kelas negatif. Menurut Li dkk (2008) beberapa penelitian menggunakan ukuran *g-mean* untuk mengevaluasi performansi dari algoritma pada masalah data *imbalance* karena ukuran ini mengkombinasikan *sensitivity* dan *specificity* dengan mengambil rata-rata geometriknya.

$$g - mean = \sqrt{sensitivity \times specificity} \tag{2.50}$$

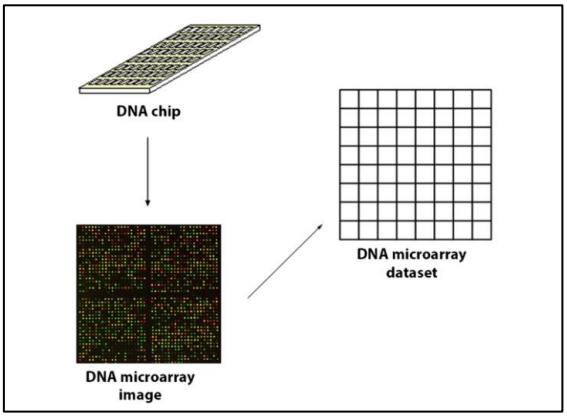
2.8 Ekspresi Gen dan Data *Microarray*

Ekspresi gen merupakan suatu proses yang mana informasi dari suatu gen digunakan dalam perpaduan dari sebuah produk gen fungsional. Produk-produk ini seringkali berupa protein, namun untuk gen seperti rRNA atau tRNA, produknya berupa suatu RNA struktural. Gen membuat protein dalam dua langkah: DNA ditranskripsi menjadi mRNA dan kemudian mRNA diterjemahkan menjadi protein.

Data *microarray* merupakan suatu gambar, yang harus ditransformasi ke dalam matriks ekspresi gen dimana baris merepresentasikan gen dan kolom merepresentasikan sampel seperti sebuah jaringan, atau suatu kondisi

eksperimental tertentu. Proses klasifikasi suatu penyakit berdasarkan *microarray* yaitu dengan mengambil suatu data sampel ekspresi gen yang terlabeli dan kemudian membangun sebuah model *classifier* yang mengklasifikasikan data sampel baru ke dalam suatu penyakit yang berbeda (Lavanya dkk, 2014).

Klasifikasi data *microarray* menimbulkan tantangan serius bagi teknik komputasi, karena dimensi yang besar (hingga beberapa puluhan ribu gen) dengan ukuran sampel yang kecil. Masalah umum dalam data *microarray* adalah yang disebut masalah ketidakseimbangan kelas. Hal ini terjadi ketika sebuah dataset didominasi oleh kelas utama atau kelas yang telah secara signifikan lebih banyak contoh dari kelas langka/ minoritas lainnya dalam data. Biasanya, orang memiliki lebih minat belajar kelas langka. Misalnya, dalam domain, kelas kanker cenderung jarang daripada kelas non-kanker karena biasanya ada lebih banyak pasien yang sehat. Namun, penting bagi praktisi untuk memprediksi dan mencegah munculnya kanker. Dalam kasus ini standar *learning* dari suatu algoritma *classifier* memiliki bias terhadap kelas yang jumlahnya lebih banyak (mayor), karena *classifier* akan menghitung akurasi berdasarkan kelas mayor sedangkan kelas minor akan cenderung diabaikan dan dianggap *noise*. Oleh karena itu pengamatan pada kelas minor lebih sering salah diklasifikasikan (Canedo dkk, 2014)



Gambar 2.10 Proses dalam Memperoleh Data *Microrray* (Canedo dkk, 2014)

(halaman ini sengaja dikosongkan)

BAB 3
METODOLOGI PENELITIAN

Pada bab ini akan dibahas terkait tahapan-tahapan yang akan dilakukan dalam penelitian ini baik meliputi tahapan dalam melakukan studi simulasi, dan penerapan pada data publik *microarray*. Sebelumnya, akan terlebih dahulu diberikan tahapan-tahapan dalam membangun *classifier* AdaBoost-SVM dan SMOTEBoost-SVM yang nantinya akan digunakan dalam klasifikasi di dalam studi simulasi maupun penerapan pada data publik *microarray*.

3.1 Tahapan Dalam Membangun *Classifier* AdaBoost-SVM

Tahapan dalam membangun model AdaBoost-SVM adalah sebagai berikut:

i. Memboboti setiap pengamatan pada data *training* dengan bobot $D_1(i) = 1/m$, dimana *m* adalah banyaknya pengamatan di dalam data *training*.
ii. Menentukan jumlah iterasi maksimal *boosting* yaitu 1-20 iterasi.
iii. Mengambil sampel sebanyak *m* tanpa pengembalian di iterasi pertama dan dengan pengembalian di iterasi selanjutnya menggunakan bobot pengamatan sebagai probabilitas terpilihnya sampel.
iv. Melatih SVM pada data sampel menggunakan Kernel linier dan Kernel RBF menggunakan sampel-sampel pengamatan yang dibentuk pada tahap (iii). Tahapan *training* SVM sebagai berikut:
    1. Mengoptimasi fungsi tujuan pada persamaan (2.32) menggunakan kuadratik *programming*, dengan fungsi kendala pada persamaan (2.33) untuk $\hat{\boldsymbol{\alpha}}$ memperoleh $\hat{\mathbf{w}}$ dan $\hat{b}$.
    2. Mengklasifikasikan seluruh pengamatan pada data *training* berdasarkan fungsi pemisah yang diperoleh.

v. Menghitung *error* dari hasil klasifikasi data *training* menggunakan SVM pada tahap (iv). *Error* dihitung menggunakan persamaan (2.38). Bila diperoleh nilai *error* $e_t = 0$ maka iterasi berhenti.

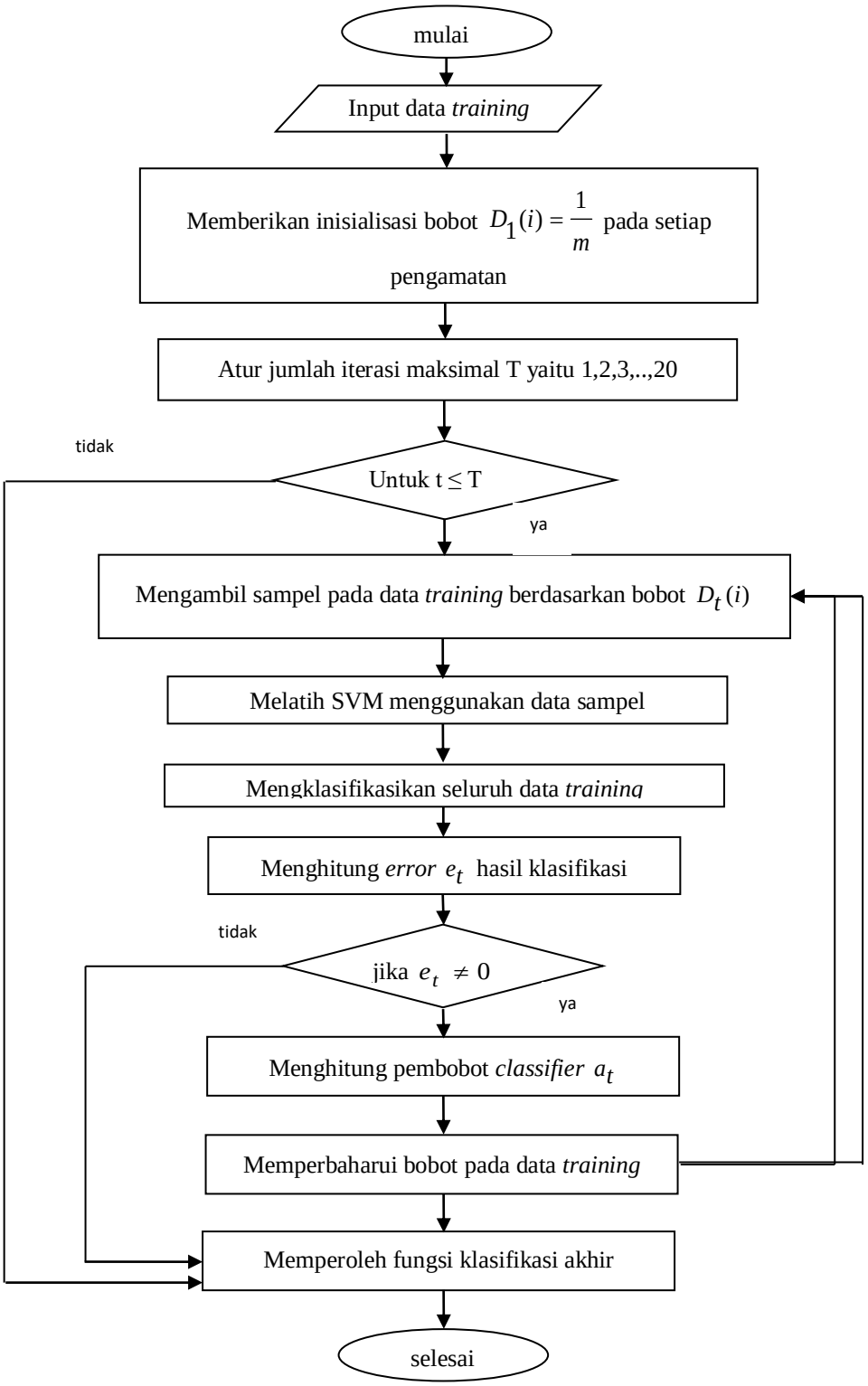
vi. Menghitung akurasi klasifikasi atau pembobot hipotesis *boosting* a_t. Perhitungan berdasarkan persamaan (2.39).

vii. Memperbaharui bobot $/$dari setiap pengamatan pada data *training* menggunakan persamaan (2.40).

viii. Mengulangi tahap (iii) sampai (vii) sampai iterasi maksimal tercapai.

ix. Menormalisasi bobot *classifier* hingga penjumlahannya menjadi 1. Kemudian diperoleh fungsi klasifikasi akhir seperti pada persamaan (2.41).

Tahapan dalam membangun AdaBoost-SVM dapat dilihat lebih detil pada Gambar 3.1.



Gambar 3.1 Tahapan AdaBoost-SVM

3.2 Tahapan Dalam Membangun *Classifier* SMOTEBoost-SVM

Tahapan dalam membangun model SMOTEBoost-SVM adalah sebagai berikut:

i. Memboboti setiap pengamatan pada data *training* dengan bobot $D_1(i) = 1/m$, dimana *m* adalah banyaknya pengamatan di dalam data *training*.

ii. Menentukan jumlah iterasi maksimal *boosting* yaitu 1-20 iterasi.

iii. Memodifikasi distribusi bobot pengamatan dengan mereplikasi pengamatan kelas minor menggunakan algoritma SMOTE sehingga proporsi kelas minoritas dan mayoritas seimbang.

iv. Menormalisasi bobot pengamatan baru (bobot *training*) hingga penjumlahannya menjadi 1.

v. Mengambil sampel sebanyak *m* dengan pengembalian menggunakan bobot *training* sebagai probabilitas terpilihnya sampel.

vi. Melatih SVM pada data sampel menggunakan Kernel linier dan Kernel RBF menggunakan sampel-sampel pengamatan yang dibentuk pada tahap (iii). Tahapan *training* SVM sebagai berikut:

    1. Mengoptimasi fungsi tujuan pada persamaan (2.32) menggunakan kuadratik *programming*, dengan fungsi kendala pada persamaan (2.33) untuk $\hat{\boldsymbol{\alpha}}$ memperoleh $\hat{\mathbf{w}}$ dan $\hat{b}$.
    2. Mengklasifikasikan seluruh pengamatan pada data *training* berdasarkan fungsi pemisah yang diperoleh.

vii. Menghitung *error* dari hasil klasifikasi data *training* menggunakan SVM pada tahap (iv). *Error* dihitung menggunakan persamaan (2.42). Bila diperoleh nilai *error* $e_t = 0$ maka iterasi berhenti.

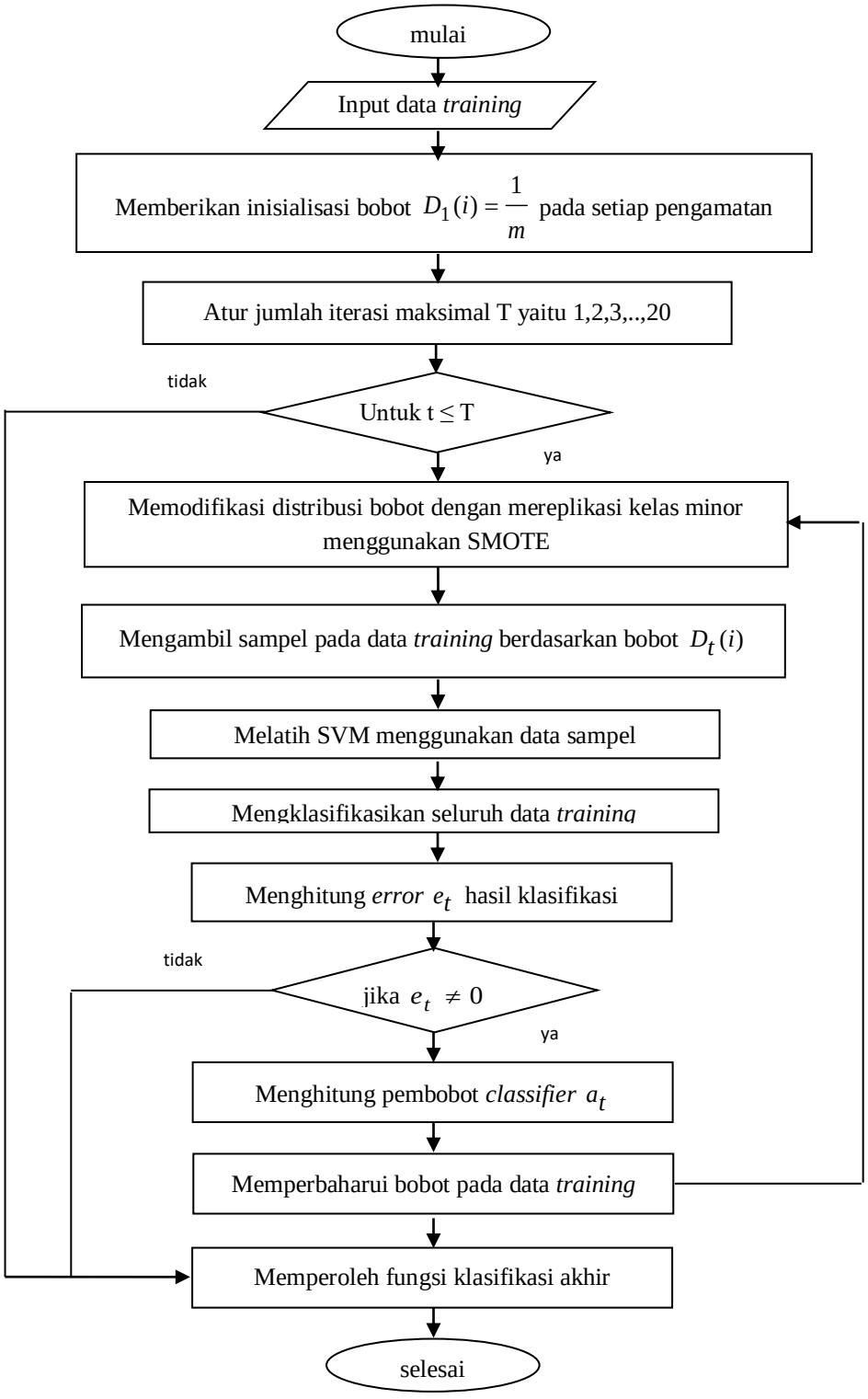
viii. Menghitung akurasi klasifikasi atau pembobot hipotesis *boosting* a_t. Perhitungan berdasarkan persamaan (2.43).

ix. Memperbaharui bobot $/$dari setiap pengamatan pada data *training* menggunakan persamaan (2.44).

x. Mengulangi tahap (iii) sampai (ix) sampai iterasi maksimal tercapai.

xi. Bobot *classifier* dinormalisasi hingga penjumlahannya sama dengan 1. Kemudian diperoleh fungsi klasifikasi akhir seperti pada persamaan (2.45).

Tahapan dalam membangun SMOTEBoost-SVM dapat dilihat lebih detil pada Gambar 3.2.



Gambar 3.2 Tahapan SMOTEBoost-SVM

3.3 Tahapan pada Studi Simulasi

Studi simulasi dilakukan untuk melihat performansi algoritma SMOTEBoost dengan menggunakan SVM sebagai komponen *classifier*, yang kemudian dibandingkan dengan performansi SVM, SMOTE-SVM dan SMOTEBoost-SVM dalam melakukan klasifikasi pada data *microarray* dengan beberapa tingkatan rasio kelas *imbalance* yang berbeda. Data simulasi didesain dengan 5 skenario tingkatan rasio *imbalance* dimana pada skenario 1, yaitu rasio antara kelas minor dan mayor adalah *balance.* Skenario 2 yaitu rasio *imbalance* antara kelas mayor dan minor adalah 2, skenario 3 yaitu dengan rasio *imbalance* 5, skenario 4 yaitu dengan rasio *imbalance* 10 dan skenario 5 yaitu dengan rasio *imbalance* 15. Tahapan dalam pembangkitan data simulasi dan klasifikasi data simulasi adalah sebagai berikut:

a. Mendesain data simulasi dengan rasio kelas minor dan mayor yaitu sebagaimana telah disebutkan sebelumnya. Pada penelitian ini hanya dikaji klasifikasi biner sehingga jumlah kelas yang akan dibangkitkan adalah 2. Jumlah gen yang akan dibangkitkan yaitu sebesar 2000 gen (fitur). Sementara jumlah sampel yang akan dibangkitkan yaitu hanya 100. Hal ini telah memenuhi karakteristik data *microarray* dimana jumlah fitur > jumlah pengamatan atau sampel (*high-dimensional* data). Matriks korelasi yang akan digunakan dalam mengukur hubungan antar fitur akan diambil dari matriks korelasi suatu *dataset microarray* publik, sehingga pada penelitian ini digunakan data kanker colon oleh Alon (1999) sebagai acuan dalam mengukur hubungan antar fitur dikarenakan memiliki jumlah fitur yang sama yaitu 2000 fitur. Metodologi dalam desain simulasi ini mengikuti eksperimen Lin dan Chen (2012). Tahapan dalam pembangkitan data pada setiap skenario rasio *imbalance* yaitu:
    i. Membangkitkan data untuk kelas mayor berdistribusi normal multivariat berdimensi 2000 dengan vektor mean **0**. Matriks varians kovarians yang digunakan yaitu berdasarkan matriks korelasi **R** dengan elemen diagonal matriks yaitu vektor standar deviasi **1**.

ii. Membangkitkan data untuk kelas minor berdistribusi normal multivariat berdimensi 2000 dengan vektor mean **0** untuk gen-gen yang non informatif dan **1** untuk gen-gen informatif. 100 gen dipilih sebagai gen-gen informatif.

iii. Mengulangi tahapan i dan ii dengan 20 ulangan.

b. Melakukan seleksi fitur dengan FCBF.

c. Menentukan *range* parameter *C* dan parameter kernel RBF γ yang akan dioptimasi dengan *grid search*. Parameter C diatur antara 0.1, 1, 10, 100 dan parameter γ antara 0.01, 0.1, 1, 10.

d. Membagi data ke dalam data *training* dan data *testing* dengan menggunakan 5-*fold cross validation* dengan stratifikasi.

e. Membangun model SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM pada data tiap *training* dengan suatu parameter yang ada pada bagian (c) hingga memperoleh fungsi pemisah yang optimum. Tahapan dalam SMOTE-SVM adalah sebagai berikut:

i. Membangkitkan data *synthetic* untuk menyeimbangkan komposisi kelas mayor dan kelas minor pada setiap data *training* menggunakan algoritma SMOTE. Parameter replikasi yang digunakan yaitu 100 untuk skenario 2 , 400 untuk skenario 3, 900 untuk skenario 4 dan 1400 untuk skenario 5. Lima tetangga terdekat dalam proses pembangkitan data *synthetic* digunakan pada skenario 2,3,dan 4 berdasarkan rekomendasi Chawla dkk (2002). Pada skenario 5 dipilih hanya tiga tetangga terdekat dengan mempertimbangkan ketersediaan jumlah sampel dari kelas minoritas.

ii. Membangun model SVM pada data *training* yang telah diseimbangkan. Optimasi fungsi tujuan SVM menggunakan persamaan (2.32) dengan fungsi kendala pada persamaan (2.33).

Tahapan dalam membangun model AdaBoost-SVM dijelaskan melalui algoritma pada sub bab 3.1 sementara tahapan dalam membangun model SMOTEBoost-SVM dijelaskan pada sub bab 3.2.

f. Mengklasifikasikan pengamatan pada data *testing* menggunakan fungsi-fungsi klasifikasi dari masing-masing model yang diperoleh pada tahapan (e).

g. Membentuk matriks konfusi dan menghitung performansi klasifikasi berdasarkan ukuran *g-mean*, akurasi, *sensitivity*, dan *specificity*.

h. Mengulangi tahapan (e) sampai ke tahap (g) untuk validasi kedua, ketiga, keempat, dan kelima. Kemudian menghitung nilai rata-rata *g-mean*, akurasi, *sensitivity*, dan *specificity*.

i. Mencari parameter C dan γ optimum dengan mengulangi tahap (e) sampai ke tahap (h) dengan menggunakan seluruh parameter C dan γ. Model dengan parameter optimum diperoleh dengan melihat nilai rata-rata *g-mean* terbesar.

j. Membandingkan performansi keempat model berdasarkan ukuran akurasi, s*ensitivity*, *specificity*, dan g-mean.

3.4. Tahapan pada Klasifikasi Data Publik *Microarray*

Data *microarray* yang digunakan dalam penelitian ini yaitu data publik yang telah dipublikasikan pada penelitian-penelitian sebelumnya. Data di-*download* melalui *package datamicroarray* dari *software* R dengan alamat (https://github.com/ramhiser/datamicroarray). Berikut deskripsi data *microarray* yang digunakan dalam penelitian ini:

Tabel 3.1 Deskripsi Data

Data	Sampel	Fitur	Distribusi Kelas		RI*
			Mayor	Minor	
Kanker Colon	62	2000	40	22	1,82
Myeloma	173	12625	173	36	3,81

*RI= *Ratio Imbalance* yaitu jumlah kelas mayor dibagi jumlah kelas minor

1. Data Kanker Colon

   Data Kanker Colon yang digunakan bersumber dari Alon (1999). Data terdiri dari 62 pasien yang terbagi ke dalam dua kelas, yakni 22 pasien berada pada kelas biopsi normal (n) dan 40 pasien berada pada kelas biopsi tumor. Masing-masing sampel terdiri atas 2905 gen yang dilabeli (t).

2. Data Myeloma

Data myeloma bersumber dari Tian (2003) terdiri dari 173 pasien yang terdiagnosis *multiple* myeloma. Data terbagi menjadi dua kelas yakni 137 pasien berada pada kelas 1 (tidak terdeteksi luka pada tulang) dan 36 pasien berada pada kelas 2 (terdeteksi luka pada tulang). Masing masing sampel terdiri atas 12625 variabel atau fitur dimana 10000 diantaranya merupakan ekspresi gen.

Selanjutnya akan dilakukan klasifikasi algoritma SVM, SMOTE-SVM, AdaBoost-SVM, dan SMOTEBoost-SVM dengan fitur seleksi FCBF pada kedua data. Langkah-langkahnya adalah sebagai berikut:

a. Melakukan *preprocessing* data dengan melakukan pengecekan *missing value* dan *scaling* data.
b. Melakukan seleksi fitur menggunakan metode FCBF.
c. Menentukan range parameter *C* dan parameter kernel RBF γ yang akan dioptimasi dengan *grid search*. Parameter C diatur antara 0.1, 1, 10, 100 dan parameter γ antara 0.01, 0.1, 1, 10.
d. Membagi data ke dalam data *training* dan data *testing* dengan menggunakan 5-*fold cross validation* dengan stratifikasi dimana komposisi dari masing-masing *fold* berisi 20% dari jumlah data mayor dan 20% dari jumlah data minor. Ilustrasi proses validasi untuk salah satu data *microarray* akan ditunjukan sebagai berikut: pada data *Breast Cancer* diketahui jumlah kelas mayor yaitu 111, dan jumlah kelas minor yaitu 57. Kemudian dibentuk 5 *fold* untuk masing-masing kelas sehingga kelima *fold* untuk kelas mayor berisi 22, 22, 22, 22, 23 pengamatan dan kelima *fold* untuk kelas minor masing-masing berisi 11, 11, 11, 12, 12 pengamatan. Proses pemilihan anggota *fold* dilakukan dengan acak dan pengamatan-pengamatan disetiap *fold* tidak tumpang tindih.

Tabel 3.2 Ilustrasi Proses Validasi

Validasi	*Fold*				
	1	2	3	4	5
1	*Testing* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Training* (k_1=12, k_2=22)	*Training* (k_1=12, k_2=23)
2	*Training* (k_1=11, k_2=22)	*Testing* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Training* (k_1=12, k_2=22)	*Training* (k_1=12, k_2=23)
3	*Training* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Testing* (k_1=11, k_2=22)	*Training* (k_1=12, k_2=22)	*Training* (k_1=12, k_2=23)
4	*Training* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Testing* (k_1=12, k_2=22)	*Training* (k_1=12, k_2=23)
5	*Training* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Training* (k_1=11, k_2=22)	*Training* (k_1=12, k_2=22)	*Testing* (k_1=12, k_2=23)

k_1= jumlah pengamatan kelas minor; k_2= jumlah pengamatan kelas mayor

Berdasarkan tabel 3.2, pada validasi pertama *fold* pertama digunakan sebagai data *testing* dan gabungan *fold* kedua, ketiga, keempat, dan kelima digunakan sebagai data *training* sehingga jumlah pengamatan pada data *testing* dan *training* pada validasi pertama masing-masing yaitu 33 dan 135 pengamatan.

e. Membangun model SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM pada data tiap *training* dengan suatu parameter yang ada pada bagian (c) hingga memperoleh fungsi pemisah yang optimum. Tahapan dalam membangun model SMOTE-SVM adalah sebagai berikut:

   i. Membangkitkan data *synthetic* untuk menyeimbangkan komposisi kelas mayor dan kelas minor pada setiap data *training* menggunakan algoritma SMOTE. Parameter replikasi yang digunakan yaitu 100 untuk data kanker colon dan 300 untuk data myeloma. Lima tetangga terdekat dalam proses pembangkitan data *synthetic* digunakan berdasarkan rekomendasi Chawla dkk (2002).

   ii. Membangun model SVM pada data *training* yang telah diseimbangkan. Optimasi fungsi tujuan SVM menggunakan persamaan (2.32) dengan fungsi kendala pada persamaan (2.33).

   Tahapan dalam membangun model AdaBoost-SVM dijelaskan melalui algoritma pada sub bab 3.1 dengan jumlah iterasi berubah-ubah dari 1-20 iterasi. Sementara tahapan dalam membangun model

SMOTEBoost-SVM dijelaskan pada sub bab 3.2 dengan input k=5, jumlah iterasi berubah-ubah mulai 1-20 iterasi dan parameter replikasi yaitu 100 untuk data kanker colon dan 300 untuk data myeloma.

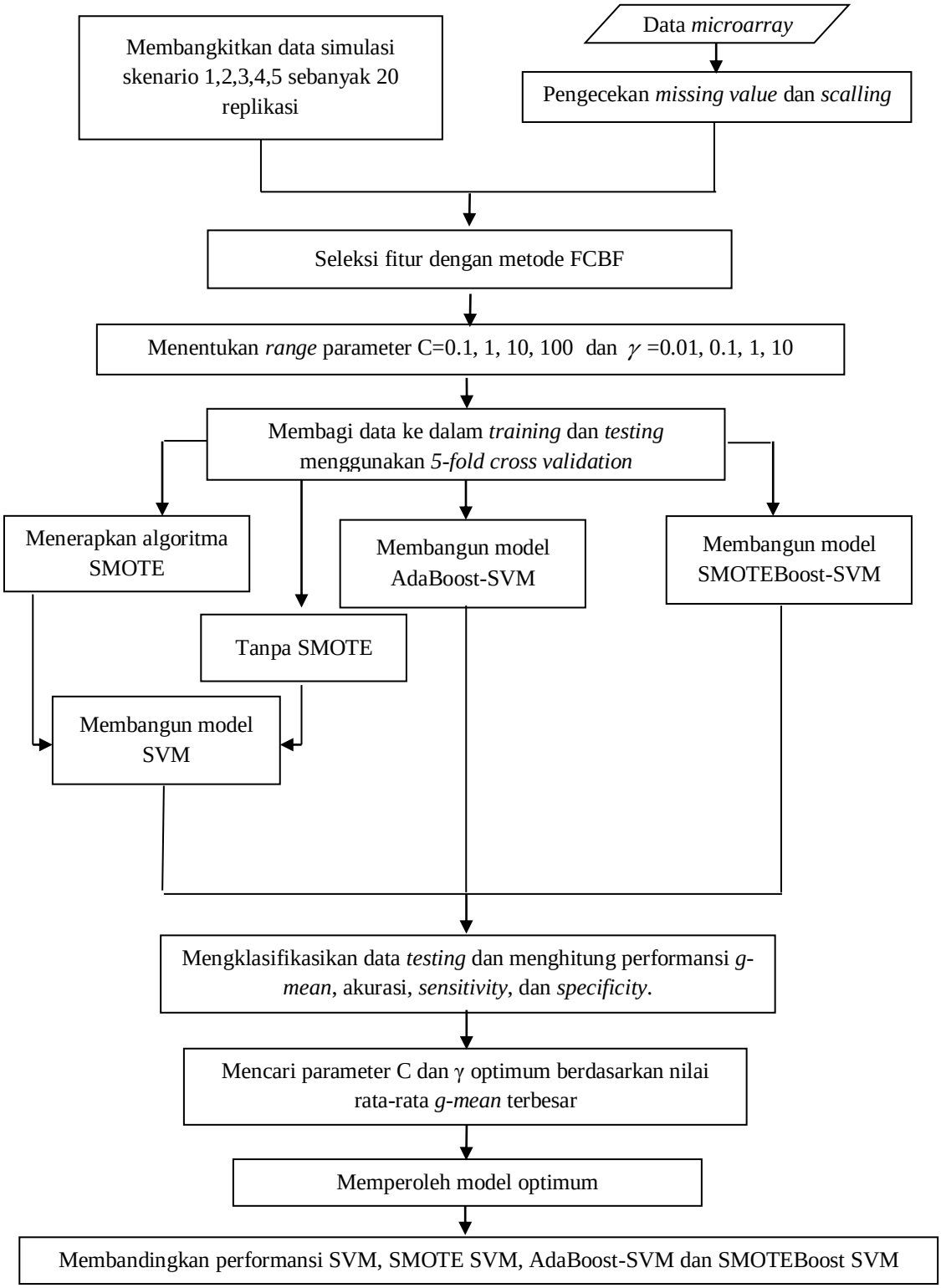
f. Mengklasifikasikan pengamatan-pengamatan pada data *testing* menggunakan fungsi klasifikasi SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM yang telah diperoleh pada tahap (e).

g. Membentuk matriks konfusi dan menghitung performansi klasifikasi berdasarkan ukuran *g-mean* akurasi, *sensitivity*, dan *specificity*.

h. Kembali ke tahap (e) sampai ke tahap (g) untuk validasi kedua, ketiga, keempat, dan kelima. Kemudian menghitung nilai rata-rata *g-mean*, akurasi, *sensitivity*, dan *specificity*.

i. Mencari parameter C dan γ optimum dengan mengulangi tahap (d) sampai ke tahap (h) dengan menggunakan tiap-tiap parameter pada bagian (c) sehingga diperoleh model optimum untuk masing masing keempat metode yang digunakan berdasarkan nilai g-mean terbesar.

j. Membandingkan performansi SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM menggunakan performansi akurasi, *sensitivity*, *specificity*, dan g-mean.



Gambar 3.3 Tahapan Penelitian

(halaman ini sengaja dikosongkan)

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai performansi hasil klasifikasi dari metode SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM pada studi simulasi dilanjutkan dengan hasil klasifikasi pada penerapan data publik *microarray*. Pada bab ini juga dilakukan evaluasi mengenai efek dari penggunaan seleksi fitur FCBF pada klasifikasi menggunakan SVM pada data publik *microarray*. Sebelumnya akan terlebih dahulu diberikan agoritma dari AdaBoost-SVM dan SMOTEBoost-SVM yang nantinya akan digunakan dalam klasifikasi.

4.1 Algoritma AdaBoost-SVM

Algoritma dari AdaBoost-SVM diberikan sebagai berikut.

***INPUT*:** sampel *training* : $\mathbf{x}_i = \{x_{i1,} x_{i2},...x_{ip}\} \in \mathbf{R}^p$, $i = 1,2,..., m$

label *training* : $y_i = \{y_1,..., y_m\} \in \{-1,+1\}$

fungsi kernel SVM

parameter C dan γ

jumlah iterasi maksimal *boosting* T

OUTPUT: *Final classifier boosting*

***BEGIN*:** 1. Memboboti setiap pengamatan pada data *training* dengan bobot

$D_1(i) = 1/m$

WHILE (t < T)

2. Membentuk suatu data *training* baru (*training set boosting)*:

IF (t=1) Mengambil sampel sebanyak *m* tanpa pengembalian berdasarkan bobot sebagai probabilitas terpilihnya sampel di iterasi pertama dan dengan pengembalian

ELSE

Mengambil sampel sebanyak *m* dengan pengembalian berdasarkan bobot sebagai probabilitas terpilihnya sampel.

END

3. Melatih *weak learner* (SVM) menggunakan data *training* baru. Proses *training* SVM dilakukan dengan melakukan optimasi pada fungsi tujuan pada persamaan (2.32) menggunakan *solver* kuadratik *programming* komersil libsvm, dengan fungsi kendala pada persamaan (2.33) untuk memperoleh $\hat{\boldsymbol{\alpha}}$, $\hat{\mathbf{w}}$ dan $\hat{b}$ yang kemudian di subtitusi ke fungsi klasifikasi $f_t(\mathbf{x})$ dan *decision function* $h_t(\mathbf{x})$.
4. Melakukan klasifikasi pada data *training* awal (input) menggunakan fungsi klasifikasi $f_t(\mathbf{x})$ yang diperoleh

   $$\hat{f}_t(\mathbf{x}_{train}) = sign\,(h_t(\mathbf{x}_{train}))$$

   Dimana

   $$h_t(\mathbf{x}_{train}) = \sum_{i=1}^{m} \boldsymbol{\alpha}_i y_i \varphi(\mathbf{x}_{train})^T \varphi(\mathbf{x}_{train}) + \hat{b}$$

5. Menghitung e_t atau *error* dari hasil klasifikasi pada data *training* menggunakan persamaan (2.38).

   IF e_t =0 maka ***BREAK*** dan menuju tahap (8).

   IF e_t >0,5 maka menuju tahap (1).
6. Menghitung akurasi klasifikasi atau pembobot hipotesis *boosting* a_t . Perhitungan berdasarkan persamaan (2.39).
7. Memperbaharui bobot $/$dari setiap pengamatan $D_{t+1}(i)$ pada data *training* menggunakan persamaan (2.40).

END WHILE.

8. Menormalisasi nilai bobot *classifier* α_t sehingga $\sum_{t=1}^{T} a_t = 1$, lalu menggabungkan *decision function* $h_t(\mathbf{x})$ yang diperoleh pada tiap iterasi menjadi suatu *final classifier* seperti pada persamaan (2.41).

END.

Ilustrasi mengenai klasifikasi menggunakan AdaBoost-SVM dapat dilihat pada Lampiran 13.

4.2 Algoritma SMOTEBoost-SVM

Algoritma dari SMOTEBoost-SVM diberikan sebagai berikut:

***INPUT*:** sampel *training* : $\mathbf{x}_i = \{x_{i1,} x_{i2},...x_{ip}\} \in \mathbf{R}^p$, $i = 1,2,...,m$, $m = m_+ + m_-$

label *training* : $y_i = \{y_1,..., y_m\} \in \{-1,+1\}$

k =jumlah tetangga terdekat

R=persentase replikasi yang akan dilakukan pada kelas minoritas

fungsi kernel SVM

parameter C dan γ

T =jumlah iterasi maksimal *boosting*

OUTPUT: *Final classifier boosting*

***BEGIN*:** 1. Memboboti setiap pengamatan pada data *training* dengan bobot $D_1(i) = 1/m$

WHILE (t < T)

2. Melakukan replikasi pengamatan-pengamatan pada kelas minoritas sebanyak R persen menggunakan algoritma SMOTE berdasarkan k-tetangga terdekat. Pada tahap ini bobot pengamatan diikutsertakan sebagai variabel sehingga nantinya diperoleh pengamatan *training* baru berjumlah $k=m+(m_+ * R/100)$ beserta bobot-bobot baru untuk pengamatan sintetis pada kelas minoritas. m_- adalah jumlah sampel kelas mayoritas dan m_+ merupakan jumlah pengamatan kelas minoritas.
3. Melakukan normalisasi bobot sehingga $\sum_{i=1}^{k} D_1(i) = 1$
4. Membentuk suatu *training set boosting* (data *training* baru) dengan cara mengambil sampel sebanyak *k* dengan pengembalian berdasarkan bobot baru pada tahap (3) sebagai probabilitas terpilihnya sampel.
5. Melatih *weak learner* (SVM) menggunakan data *training* baru pada tahap (4). Proses *training* SVM dilakukan dengan melakukan optimasi pada fungsi tujuan pada persamaan pada

persamaan (2.32) menggunakan *solver* kuadratik *programming* komersil libsvm, dengan fungsi kendala pada persamaan (2.33) untuk memperoleh $\hat{\boldsymbol{\alpha}}$, $\hat{\mathbf{w}}$ dan $\hat{b}$ yang kemudian di subtitusi ke fungsi klasifikasi $f_t(\mathbf{x})$ dan *decision function* $h_t(\mathbf{x})$.

6. Melakukan klasifikasi pada data *training* awal (input) menggunakan fungsi klasifikasi yang diperoleh

$$\hat{f}_t(\mathbf{x}_{train}) = sign\,(h_t(\mathbf{x}_{train}))$$

Dimana

$$h_t(\mathbf{x}_{train}) = \sum_{i=1}^{m} \boldsymbol{\alpha}_i y_i \varphi(\mathbf{x}_{train})^T \varphi(\mathbf{x}_{train}) + \hat{b}$$

7. Menghitung e_t atau *error* dari hasil klasifikasi pada data *training* menggunakan persamaan (2.42).
   IF e_t =0 maka ***BREAK*** dan menuju tahap (10).
   IF e_t >0,5 maka menuju tahap (1).
8. Menghitung akurasi klasifikasi atau pembobot hipotesis *boosting* a_t . Perhitungan berdasarkan persamaan (2.43).
9. Memperbaharui bobot $/$dari setiap pengamatan $D_{t+1}(i)$ pada data *training* awal (input) menggunakan persamaan (2.44).

END WHILE

10. Menormalisasi nilai bobot *classifier* α_t sehingga $\sum_{t=1}^{T} a_t = 1$, lalu Menggabungkan *decision function* $h_t(\mathbf{x})$ yang diperoleh pada tiap iterasi menjadi suatu *final classifier* seperti pada persamaan (2.45).

END.

Ilustrasi mengenai klasifikasi menggunakan SMOTEBoost-SVM dapat dilihat pada Lampiran 14.

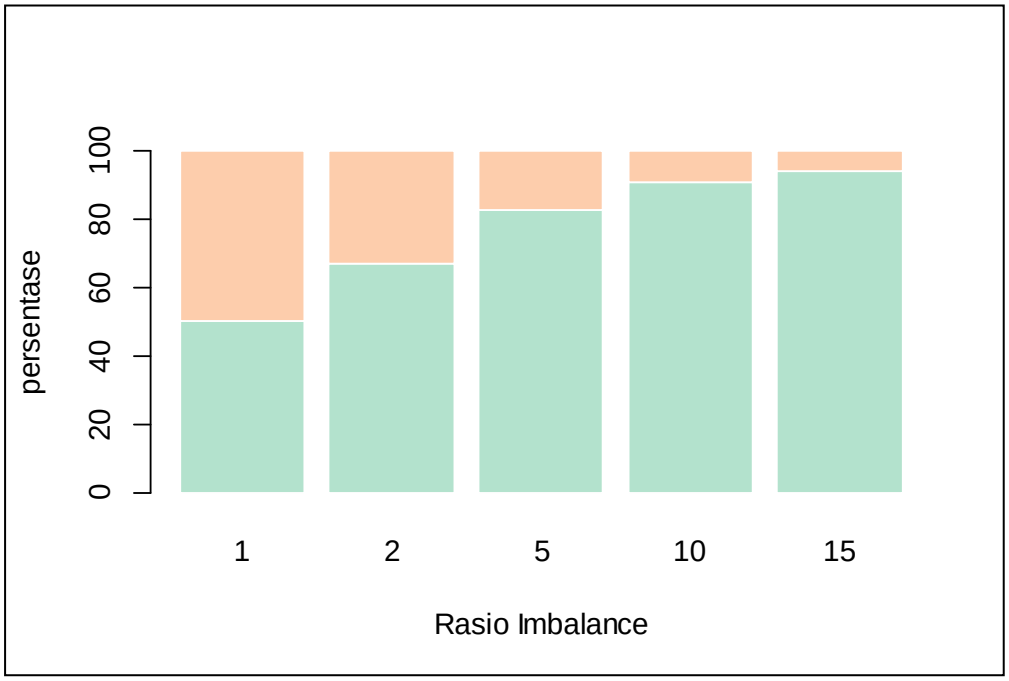
4.3 Studi Simulasi

Sebagaimana telah disebutkan pada tujuan penelitian bahwasanya studi simulasi bertujuan untuk melihat performansi dari algortima SVM, SMOTE-

SVM, AdaBoost-SVM dan SMOTEBoost-SVM pada beberapa tingkatan rasio *imbalance* yang berbeda sehingga skenario simulasi yang difokuskan pada studi simulasi ini yaitu rasio antara jumlah pengamatan kelas minoritas dan kelas mayoritas. Data simulasi dibangkitkan berdasarkan 5 skenario dimana skenario 1 yaitu Rasio kelas *imbalance* 1 (kondisi *balance*), skenario 2 yaitu rasio *imbalance* 2, skenario 3 yaitu 5, skenario 4 yaitu 10 dan skenario 5 yaitu 15. Jumlah sampel yang dibangkitkan yaitu 100. Data untuk kelas mayoritas dibangkitkan mengikuti distribusi normal multivariat dengan mean **0**, dan untuk kelas minor dibangkitkan mengikuti distribusi normal multivariat dengan mean **0** untuk gen (fitur) yang tidak informatif dan mean **1** untuk gen (fitur) yang informatif. Sebelumnya, 100 gen dipilih secara random sebagai gen-gen yang informatif dari total 2000 gen yang dibangkitkan. Matriks varians dan kovarian yang digunakan berdasar pada nilai korelasi antar gen (fitur) pada data kanker colon oleh Alon (1999). Proses pembangkitan data direplikasi sebanyak 20 kali. Kemudian performansi klasifikasi dari SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM akan dilihat pada kelima data set bangkitan dengan menggunakan 20 replikasi.

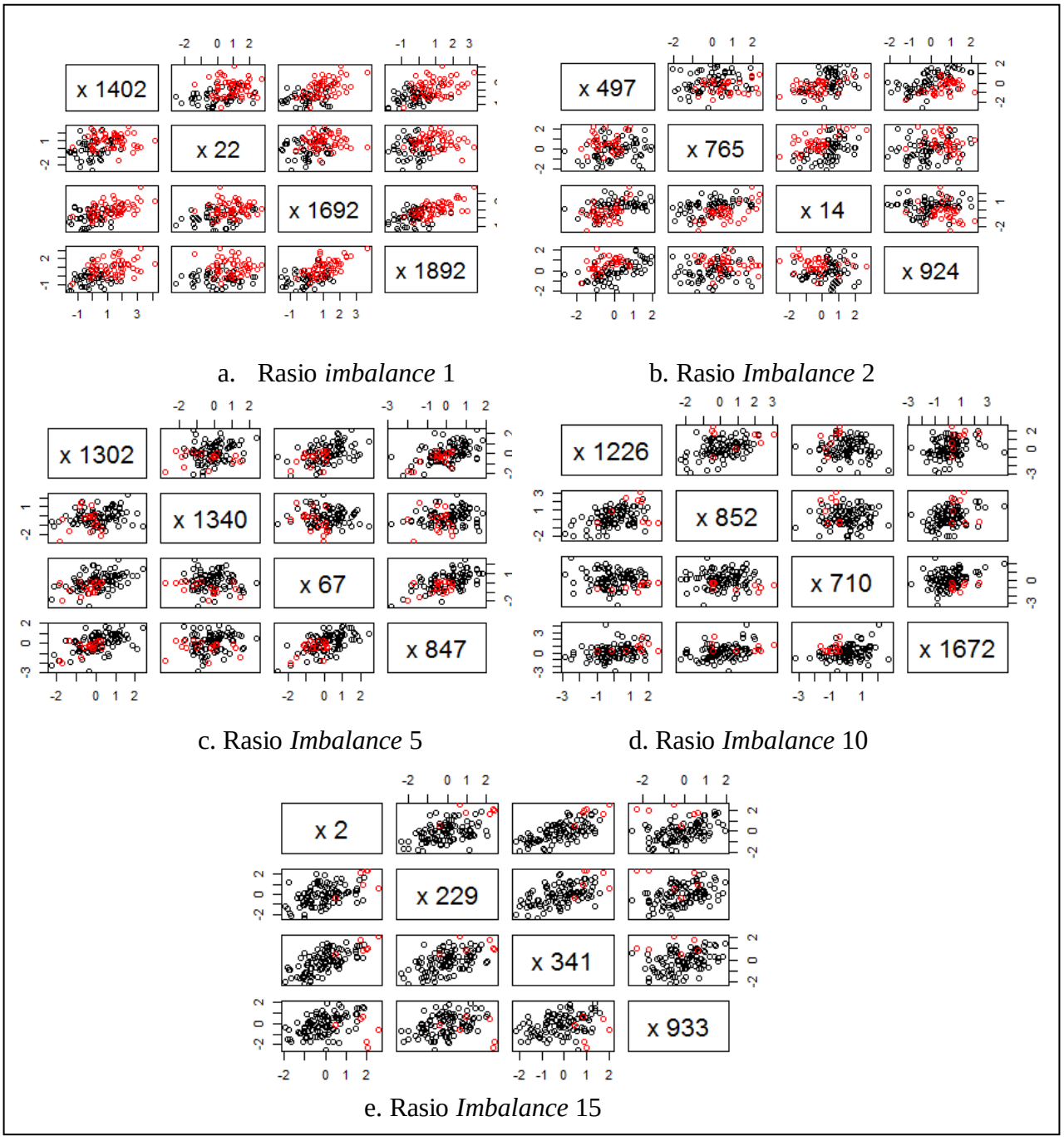
4.3.1 Karakteristik Data Simulasi

Bagian ini menyajikan karakteristik dari kelima skenario data hasil bangkitan. Setiap data tersebut memiliki karakteristik yang berbeda jika dilihat dari pola persebaran dari setiap fitur-fitur dan kategori kelasnya. Berikut disajikan persentase dari distribusi kelas pada tiap skenario data bangkitan pada Gambar 4.1. Berdasarkan Gambar 4.1 dan 4.2 tersebut dapat dilihat bahwa pada kondisi rasio tertinggi yakni *imbalance* 15, pengamatan kelas mayoritas sangat mendominasi dengan jumlah 91 pengamatan dibandingkan 6 pengamatan dari kelas minoritas. Hal ini nanti tentunya akan menyulitkan *classifier* untuk dapat mengklasifikasikan pengamatan ke kelas nya masing-masin.



Gambar 4.1 Persentase Distribusi Kelas pada Tingkatan Rasio *Imbalance* yang dibangkitkan

Selanjutnya akan disajikan persebaran data pengamatan dari beberapa fitur-fitur informatif pada tiap skenario data bangkitan replikasi pertama melalui Gambar 4.2. Berdasarkan Gambar 4.2 dapat dilihat bahwa pola persebaran data pada beberapa fitur-fitur yang informatif menunjukan pola yang sangat kompleks. Pola data juga mengindikasikan bahwa fungsi pemisah dari klasifikasinya akan berupa nonlinier sehingga akan digunakan fungsi kernel untuk memetakan ruang input ke dimensi yang lebih tinggi. Fungsi kernel linier dan radial akan digunakan dalam klasifikasi ini.



a. Rasio *imbalance* 1

b. Rasio *Imbalance* 2

c. Rasio *Imbalance* 5

d. Rasio *Imbalance* 10

e. Rasio *Imbalance* 15

Gambar 4.2 Pola Persebaran Kelas pada Tiap data bangkitan

4.3.2 Klasifikasi Pada Data Rasio *Imbalance* 1

Pada skenario ini, SVM dan AdaBoost-SVM akan digunakan untuk mengklasifikasikan data. Fungsi kernel yang digunakan yaitu kernel radial dan linier dengan range parameter yang sama untuk SVM dan AdaBoost-SVM yakni

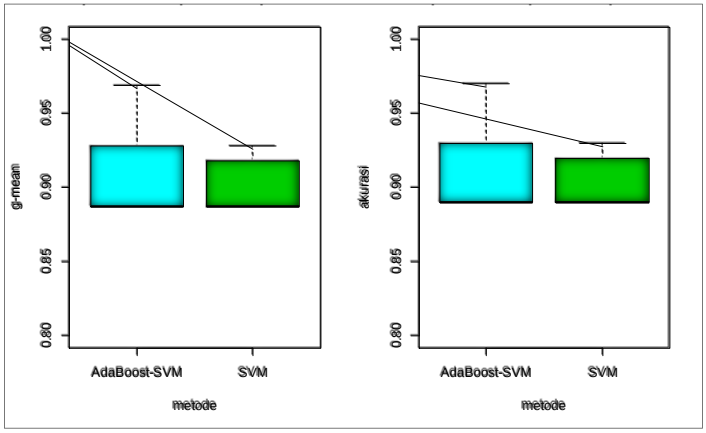
C=0,1; 1;10; 100 dan γ=0,01; 0,1; 1; 10. Kemudian parameter optimum diperoleh berdasarkan nilai rata-rata g-mean tertinggi pada 20 replikasi yang dilakukan.

Tabel 4.1 Nilai Performansi pada Klasifikasi Data Rasio *Imbalance* 1 pada Setiap Parameter

Kernel	C	γ	SVM		AdaBoost-SVM	
			G-mean	Akurasi	G-mean	Akurasi
Radial	0,1	0,01	0,8707	0,8725	0,8909	0,892
	1	0,01	0,878	0,88	**0,9018**	**0,903**
	10	0,01	**0,8895**	**0,8915**	0,8944	0,896
	100	0,01	0,8682	0,8705	0,7994	0,8443
	0,1	0,1	0,8728	0,875	0,8892	0,892
	1	0,1	0,8689	0,8705	0,8357	0,8681
	10	0,1	0,8318	0,8335	-	-
	100	0,1	0,8318	0,8335	-	-
	0,1	1	0,4481	0,6	-	-
	1	1	0,5893	0,652	-	-
	10	1	0,6115	0,6675	-	-
	100	1	0,6115	0,6675	-	-
	0,1	10	0,418	0,6595	-	-
	1	10	0,0261	0,504	-	-
	10	10	0,0922	0,5095	-	-
	100	10	0,0923	0,5095	-	-
Linier	0,1		**0,9001**	**0,9025**	**0,9078**	**0,91**
	1		0,8816	0,884	0,8747	0,8771
	10		0,8119	0,813	0,8194	0,82
	100		0,8098	0,812	-	-

Tabel 4.1 menunjukan nilai g-mean dan akurasi pada klasifikasi data *balance* menggunakan SVM dan AdaBoost-SVM dengan beberapa kombinasi parameter C dan γ. hasil g-mean dan akurasi merupakan rata-rata dari kelima *fold* validasi dalam 20 replikasi. Berdasarkan Tabel 4.1 diperoleh performansi tertinggi SVM saat menggunakan kernel radial yaitu dengan parameter C=10 dan γ=0,01 dengan nilai ketepatan klasifikasi sebesar 89,15% dan nilai g-mean sebesar 0,8895 sementara nilai performansi tertinggi SVM saat menggunakan kernel linier yaitu dengan parameter C=0,1 dengan nilai ketepatan klasifikasi sebesar 90,01% dan g-mean sebesar 0,9025.

Performansi tertinggi AdaBoost-SVM saat menggunakan kernel radial yaitu dengan parameter C=1 dan γ=0,01 dengan nilai ketepatan klasifikasi sebesar 90,3% dan nilai g-mean sebesar 0,9018 sementara nilai performansi tertinggi AdaBoost-SVM saat menggunakan kernel linier yaitu dengan parameter C=0,1 dengan nilai ketepatan klasifikasi sebesar 91% dan g-mean sebesar 0,9078. Berdasarkan Tabel 4.1 juga dapat dilihat bahwa AdaBoost tidak dapat bekerja di beberapa parameter dikarenakan tidak dapat mengambil keuntungan dari kesalahan klasifikasi dari *base classifier* SVM di iterasi pertama (nilai error training = 0 di iterasi pertama).



Gambar 4.3 *Boxplot* Performansi Klasifikasi Data Rasio *Balance* Menggunakan Parameter Optimum

Tabel 4.2 *Summary* Performansi Klasifikasi Data Rasio *Balance* Menggunakan Parameter Optimum

Metode	G-mean				Akurasi			
	Min	Median	Mean	Max	Min	Median	Mean	Max
SVM	0,8872	0,8872	0,9001	0,9281	0,89	0,89	0,9025	0,93
AdaBoost-SVM	0,8872	0,8872	0,9078	0,9692	0,89	0,89	0,91	0,97

Gambar 4.3 dan Tabel 4.2 menunjukkan performansi g-mean dan akurasi pada parameter optimum pada 20 replikasi. Berdasarkan Gambar 4.3 dan Tabel 4.2 dapat dilihat bahwa performansi AdaBoost-SVM cenderung menghasilkan performansi g-mean dan akurasi lebih baik dibandingkan SVM pada 20 replikasi namun meskipun demikian variasi nilai-nilai g-mean dan akurasi dari 20 replikasi

yang di hasilkan AdaBoost-SVM lebih besar dibandingkan SVM. dalam hal ini SVM dapat dikatakan lebih stabil pada klasifikasi data rasio *balance*.

4.3.3 Klasifikasi Pada Data Rasio *Imbalance* 2

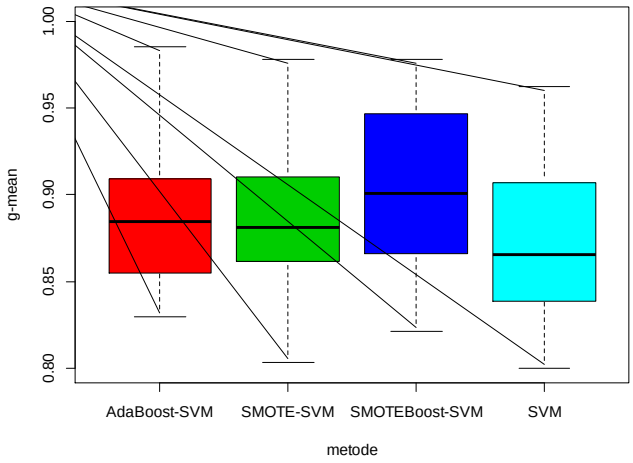
Pada skenario ini, SVM, SMOTE-SVM, AdaBoost-SVM, dan SMOTEBoost-SVM akan digunakan untuk mengklasifikasikan data. Fungsi kernel yang digunakan yaitu kernel radial dan linier dengan *range* parameter yang sama untuk keempat metode yang digunakan yakni C=0,1; 1;10; 100 dan γ=0,01; 0,1; 1; 10.

Tabel 4.3 Nilai Performansi Pada Klasifikasi Data Rasio *Imbalance* 2 Pada Setiap Parameter

Kernel	C	γ	Rata-Rata G-mean 20 Replikasi			
			SVM	SMOTE-SVM	AdaBoost-SVM	SMOTEBoost-SVM
Radial	0,1	0,01	0,0000	0,6262	0,6661	0,8604
	1	0,01	0,8272	0,8586	**0,8850**	**0,9024**
	10	0,01	0,8678	**0,8856**	0,7356	0,8921
	100	0,01	**0,8692**	0,8629	0,7729	0,8832
	0,1	0,1	0,0000	0,7561	0,7309	0,8659
	1	0,1	0,8456	0,8651	0,7774	0,8689
	10	0,1	0,8354	0,8406	-	0,8619
	100	0,1	0,8353	0,8363	-	0,8637
	0,1	1	0,0000	0,0000	0,0000	0,0191
	1	1	0,0000	0,0808	-	0,4551
	10	1	0,0236	0,1125	-	0,4905
	100	1	0,0236	0,1093	-	0,4860
	0,1	10	0,0000	0,0000	0,0000	0,0081
	1	10	0,0000	0,0000	-	0,0000
	10	10	0,0000	0,0000	-	0,0000
	100	10	0,0000	0,0000	-	0,0000
Linier	0,1		0,8643	**0,8814**	**0,8876**	**0,8873**
	1		**0,8721**	0,8780	0,8700	0,8608
	10		0,8693	0,8746	0,8613	0,8542
	100		0,8608	0,8670	0,8405	0,8475

Tabel 4.3 menunjukan nilai rata-rata g-mean dari 20 replikasi pada klasifikasi data rasio *imbalance* sama dengan 2 menggunakan SVM, SMOTE-SVM, AdaBoost-SVM, dan SMOTEBoost-SVM dengan beberapa kombinasi parameter C dan γ.

Iterasi optimum dalam *boosting* diperoleh berdasarkan nilai g-mean tertinggi dalam *trial error* 1-20 iterasi maksimal. Kemudian parameter optimum diperoleh berdasarkan nilai rata-rata g-mean tertinggi pada 20 replikasi yang dilakukan. Hasil g-mean yang ditampilkan merupakan rata-rata dari kelima *fold* validasi dalam 20 replikasi. Berdasarkan Tabel 4.3 dapat dilihat bahwa pada fungsi kernel radial, SMOTEBoost-SVM menghasilkan nilai g-mean tertinggi jika dibandingkan ketiga metode lainnya yakni sebesar 0,9024 dengan parameter C=1 dan γ=0,01. Namun pada fungsi kernel linier, AdaBoost-SVM menghasilkan nilai g-mean tertinggi yakni sebesar 0,8876 dengan parameter C=0,1.



Gambar 4.4 *Boxplot* Performansi Klasifikasi data Rasio *Imbalance* 2 Menggunakan Parameter Optimum

Tabel 4.4 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 2 Menggunakan Parameter Optimum

Metode	Min	Q1	Median	Q3	Mean	Max
SVM	0,8004	0,8412	0,8654	0,9058	0,8721	0,9625
AdaBoost-SVM	0,8295	0,8552	0,8848	0,9082	0,8876	0,9852
SMOTE-SVM	0,8037	0,8665	0,8814	0,901	0,8856	0,9779
SMOTEBoost-SVM	0,8211	0,867	0,901	0,9429	0,9024	0,9779

Gambar 4.4 dan tabel 4.4 menunjukkan performansi g-mean pada parameter optimum pada 20 replikasi. Berdasarkan gambar 4.4 dan tabel 4.4 dapat dilihat bahwa SMOTEBoost- SVM cenderung menghasilkan nilai-nilai g-mean yang lebih tinggi dibandingkan ketiga metode lain pada 20 replikasi. Berdasarkan

Gambar 4.4 dan Tabel 4.4 dapat dilihat pula bahwa variasi nilai-nilai g-mean yang dihasilkan SMOTE-SVM lebih kecil dibandingkan ketiga metode yang lain, hal ini berarti bahwa nilai-nilai g-mean yang dihasilkan SMOTE-SVM pada parameter optimum dalam 20 replikasi cenderung lebih stabil dibandingkan ketiga metode yang lain. Berdasarkan Tabel 4.4 Adaboost-SVM menghasilkan nilai maksimum dan nilai- minimum g-mean tertinggi dibandingkan ketiga metode lain yakni 0,9852 pada 20 replikasi yang digunakan.

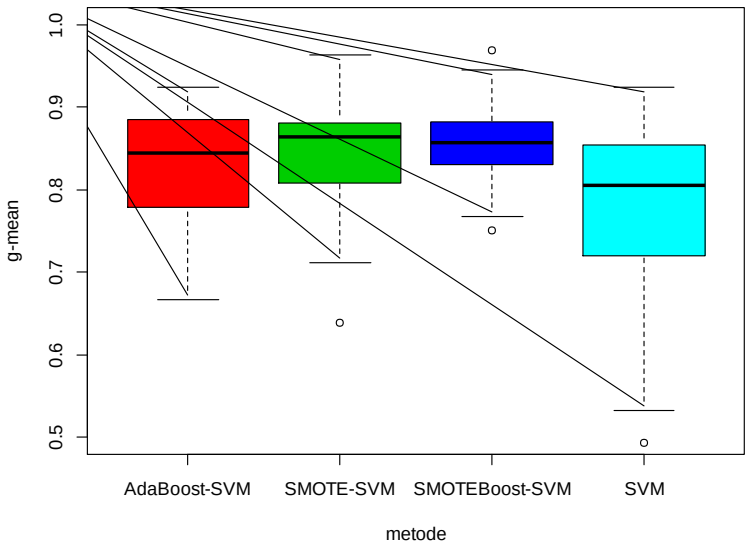
4.3.4 Klasifikasi Pada Data Rasio *Imbalance* 5

Pada skenario ini, SVM, SMOTE-SVM, AdaBoost-SVM, dan SMOTEBoost-SVM akan digunakan untuk mengklasifikasikan data.

Tabel 4.5 Nilai Performansi pada Klasifikasi Data Rasio *Imbalance* 5 pada Setiap Parameter

Kernel	C	γ	Rata-Rata G-mean 20 Replikasi			
			SVM	SMOTE-SVM	AdaBoost-SVM	SMOTEBoost-SVM
Radial	0,1	0,01	0,0000	0,7371	0,1571	0,7934
	1	0,01	0,2660	**0,8445**	**0,8258**	**0,8566**
	10	0,01	**0,7477**	0,8333	0,8104	0,8242
	100	0,01	0,7326	0,7707	0,7437	0,7643
	0,1	0,1	0,0000	0,8108	0,5702	0,8018
	1	0,1	0,6508	0,7931	0,7467	0,6644
	10	0,1	0,6971	0,7206	0,6177	0,7047
	100	0,1	0,7026	0,7184	-	0,7393
	0,1	1	0,0000	0,0000	0,0000	0,0590
	1	1	0,0000	0,0474	-	0,2146
	10	1	0,0058	0,0677	-	0,1764
	100	1	0,0058	0,0749	-	0,1936
	0,1	10	0,0000	0,0000	0,0000	0,0000
	1	10	0,0000	0,0000	-	0,0098
	10	10	0,0000	0,0000	-	0,0171
	100	10	0,0000	0,0000	-	0,0122
Linier	0,1		0,7454	**0,8447**	**0,7535**	0,8515
	1		**0,7756**	0,8404	0,7479	0,8525
	10		0,7579	0,8142	0,7056	0,8532
	100		0,7547	0,8057	0,6875	**0,8562**

Fungsi kernel yang digunakan yaitu kernel radial dan linier dengan range parameter yang sama untuk keempat metode yang digunakan yakni C=0,1; 1;10; 100 dan γ=0,01; 0,1; 1; 10. Iterasi optimum dalam *boosting* diperoleh berdasarkan nilai g-mean tertinggi dalam *trial error* 1-20 iterasi maksimal. Kemudian parameter optimum diperoleh berdasarkan nilai rata-rata g-mean tertinggi pada 20 replikasi yang dilakukan. Tabel 4.5 menunjukkan nilai rata-rata g-mean dari 20 replikasi pada klasifikasi data rasio *imbalance* sama dengan 5 menggunakan SVM, SMOTE-SVM, AdaBoost-SVM, dan SMOTEBoost-SVM dengan beberapa kombinasi parameter C dan γ. Hasil g-mean yang ditampilkan merupakan rata-rata dari kelima *fold* validasi dalam 20 replikasi. Berdasarkan Tabel 4.5 dapat dilihat bahwa pada fungsi kernel radial, SMOTEBoost-SVM menghasilkan nilai g-mean tertinggi sebesar 0,8566 parameter C=1 dan γ=0,01 jika dibandingkan ketiga metode lainnya dengan. SMOTEBoost-SVM juga menghasilkan nilai g-mean tertinggi pada fungsi kernel linier yakni sebesar 0,8562 dengan parameter C=100.



Gambar 4.5 *Boxplot* Performansi Klasifikasi Data Rasio *Imbalance* 5 Menggunakan Parameter Optimum

Tabel 4.6 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 5 Menggunakan Parameter optimum

	G-mean					
Metode	Min	Q1	Median	Q 3	Mean	Max
SVM	0,4933	0,7209	0,8054	0,8522	0,7756	0,9242
AdaBoost-SVM	0,6667	0,7847	0,844	0,8834	0,8258	0,9242
SMOTE-SVM	0,6388	0,8131	0,864	0,8772	0,8447	0,963
SMOTEBoost-SVM	0,7512	0,8304	0,8567	0,8811	0,8566	0,969

Gambar 4.5 dan Tabel 4.6 menunjukkan performansi g-mean pada parameter optimum pada 20 replikasi. Berdasarkan Gambar 4.5 dan Tabel 4.6 dapat dilihat bahwa SMOTEBoost- SVM cenderung menghasilkan nilai-nilai g-mean yang lebih tinggi dibandingkan ketiga metode lain pada 20 replikasi yang digunakan. Lebih jauh lagi, pada Gambar 4.5 juga dapat lihiat bahwa SMOTEBoost-SVM menghasilkan variasi nilai-nilai g-mean yang lebih kecil dibandingkan ketiga metode yang lain sehingga dapat dikatakan bahwa SMOTEBoost-SVM cenderung menghasilkan nilai-nilai g-mean yang lebih stabil dalam 20 replikasi yang digunakan.

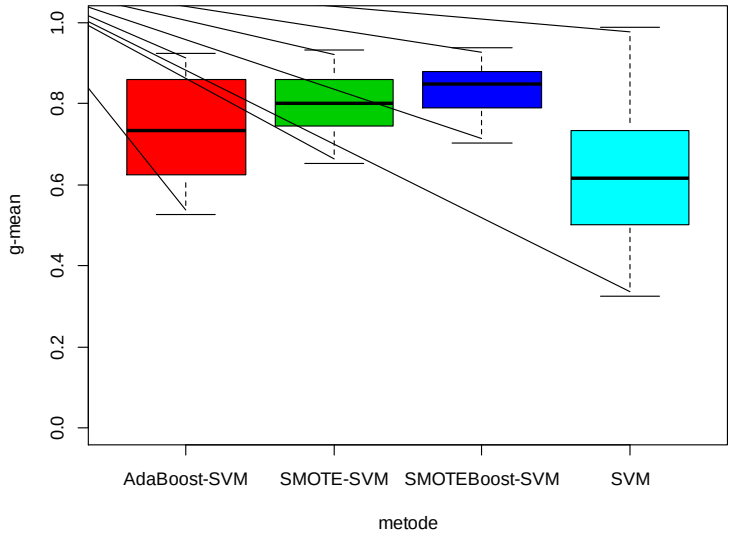
4.3.5 Klasifikasi Pada Data Rasio *Imbalance* 10

Pada skenario ini, data dibangkitkan dengan rasio pengamatan kelas mayor dan minor sebesar 10 kali sehingga pengamatan kelas minor akan berjumlah 9 pengamatan dan pengamatan pada kelas mayor akan berjumlah 91 pengamatan. Kemudian algoritma SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM akan digunakan untuk mengklasifikan pengamtan-pengamatan tersebut. Parameter kernel yang digunakan sama seperti sub bab sebelumnya. Performansi hasil klasifikasi untuk semua parameter dan fungsi kernel yang digunakan ditampilkan pada Tabel 4.7.

Tabel 4.7 Nilai Performansi pada Klasifikasi Data Rasio *Imbalance* 10 pada Setiap Parameter

Kernel	C	γ	Rata-Rata G-mean 20 Replikasi			
			SVM	SMOTE-SVM	AdaBoost-SVM	SMOTEBoost-SVM
Radial	0,1	0,01	0,0000	0,6829	0,0895	0,7107
	1	0,01	0,0412	**0,8016**	0,6313	**0,8329**
	10	0,01	0,3838	0,7708	**0,7452**	0,8151
	100	0,01	0,6076	0,7168	0,7068	0,7785
	0,1	0,1	0,0000	0,7810	0,4011	0,7903
	1	0,1	0,2919	0,7538	0,7107	0,7860
	10	0,1	0,6225	0,6751	0,4690	0,7422
	100	0,1	**0,6287**	0,6163	0,6110	0,7456
	0,1	1	0,0000	0,2877	0,0000	0,2907
	1	1	0,0141	0,2977	0,2756	0,4562
	10	1	0,1177	0,2885	0,0000	0,5349
	100	1	0,1206	0,2912	0,0000	0,5213
	0,1	10	0,0000	0,0024	0,0000	0,0171
	1	10	0,0000	0,0446	-	0,2010
	10	10	0,0141	0,0449	-	0,2693
	100	10	0,0141	0,0514	-	0,2239
Linier	0,1		0,2689	**0,7885**	0,7081	**0,8186**
	1		0,5463	0,7641	0,6909	0,8056
	10		**0,5775**	0,7141	0,6708	0,7983
	100		0,5762	0,7218	**0,7234**	0,7763

Hasil g-mean yang ditampilkan merupakan rataan dari kelima *fold* yang kemudian di rata-rata kan lagi untuk ke-20 replikasi yang digunakan. Dengan melihat Tabel 4.5 dapat diketahui bahwa pada fungsi kernel radial, SMOTEBoost-SVM menghasilkan nilai g-mean tertinggi sebesar 0,8329 parameter C=1 dan γ=0,01 jika dibandingkan ketiga metode lainnya. SMOTEBoost-SVM juga menghasilkan nilai g-mean tertinggi pada fungsi kernel linier yakni sebesar 0,8186 dengan parameter C=0.1. Selanjutnya akan dilihat pola performansi g-mean pada 20 replikasi melalui *box* dan *whisker plot* untuk setiap metode dengan parameter optimum



Gambar 4.6 *Boxplot* Performansi Klasifikasi Data Rasio *Imbalance* 10 Menggunakan Parameter Optimum

Berdasarkan Gambar 4.6 dapat dilihat bahwa SMOTEBoost-SVM cenderung menghasilkan nilai-nilai g-mean yang lebih tinggi di dalam 20 ulangan yang digunakan dibandingkan ketiga metode lain pada parameter optimum. SMOTEBoost-SVM juga menghasilkan variasi nilai g-mean yang lebih kecil. Hal ini ditunjukkan dengan *box* yang lebih sempit dibandingkan ketiga metode lain. Hal ini berarti bahwa SMOTEBoost menghasilkan performansi g-mean yang lebih konsisten dalam 20 replikasi yang digunakan dibandingkan ketiga metode yang lain. SVM cenderung menghasilkan nilai-nilai performansi g-mean yang lebih kecil dengan variasi nilai-nilai g-mean yang lebih besar dibandingkan ketiga metode lain dalam 20 replikasi yang digunakan. Summary nilai-nilai performansi g-mean pada 20 replikasi yang digunakan untuk keempat metode ditampilkan pada Tabel 4.8.

Tabel 4.8 *Summary* Performansi Klasifikasi Data Rasio *Imbalance* 10 Menggunakan Parameter Optimum

	G-mean					
Metode	Min	Q1	Median	Q 3	Mean	Max
SVM	0,3250	0,5126	0,6156	0,7318	0,6287	0,9887
AdaBoost-SVM	0,5250	0,6360	0,7318	0,8510	0,7452	0,9243
SMOTE-SVM	0,6516	0,7493	0,8007	0,8589	0,8016	0,9318
SMOTEBoost-SVM	0,7025	0,7914	0,8467	0,8723	0,8329	0,9376

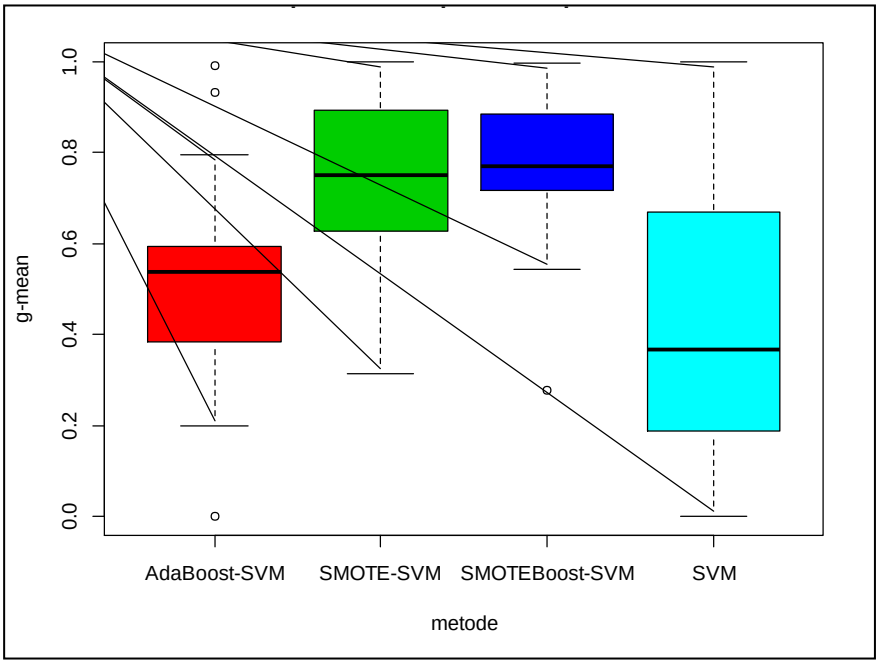
4.3.6 Klasifikasi Pada Data Rasio *Imbalance* 15

Pada bagian ini klasifikasi akan dilakukan pada data yang dibangkitkan dengan rasio *imbalance* 15 kali. Pada data ini pengamatan pada kelas minor berjumlah 6 pengamatan dan pengamatan pada kelas mayor akan berjumlah 94 pengamatan.

Tabel 4.9 Nilai Performansi pada Klasifikasi Data Rasio *Imbalance* 15 pada Setiap Parameter

Kernel	C	γ	Rata-Rata G-mean 20 Replikasi			
			SVM	SMOTE-SVM	AdaBoost-SVM	SMOTEBoost-SVM
Radial	0,1	0,01	0,0000	0,4953	0,0471	0,4939
	1	0,01	0,0000	0,7262	0,3835	**0,7725**
	10	0,01	0,2471	0,7255	0,4544	0,7590
	100	0,01	**0,3607**	0,7114	**0,5262**	0,7122
	0,1	0,1	0,0000	**0,7367**	0,2909	0,7726
	1	0,1	0,2310	0,7251	0,4856	0,7414
	10	0,1	0,3531	0,6835	0,3828	0,6915
	100	0,1	0,4211	0,6210	0,2981	0,6460
	0,1	1	0,0000	0,4319	0,1663	0,4987
	1	1	0,0865	0,4651	0,3389	0,6178
	10	1	0,2413	0,4333	0,3279	0,5169
	100	1	0,2874	0,4374	0,1500	0,6070
	0,1	10	0,0000	0,2114	0,1371	0,1929
	1	10	0,0468	0,2606	0,1589	0,5107
	10	10	0,1462	0,2522	0,0000	0,5298
	100	10	0,1462	0,2499	0,3000	0,5577
Linier	0,1		0,1880	**0,7512**	0,4814	**0,7345**
	1		0,3085	0,7377	0,4833	0,7004
	10		0,3664	0,7103	**0,5174**	0,7060
	100		**0,3758**	0,7187	0,4758	0,7004

Performansi hasil klasifikasi untuk semua parameter dan fungsi kernel yang digunakan ditampilkan pada Tabel 4.9. Dengan melihat Tabel 4.9 dapat diketahui bahwa pada fungsi kernel radial, SMOTEBoost-SVM menghasilkan nilai g-mean tertinggi sebesar 0,7725 dengan parameter C=1 dan γ=0,01 jika dibandingkan ketiga metode lainnya. Sementara pada fungsi kernel linier, SMOTE-SVM menghasilkan nilai performansi g-mean tertinggi yakni sebesar 0,7512 dengan parameter C=0.1. berdasarkan tabel 4.10 dapat dilihat pula bahwa SVM menghasilkan performansi g-mean terkecil jika dibandingkan ketiga metode lainnya baik pada fungsi kernel radial maupun linier.



Gambar 4.7 *Boxplot* Performansi Klasifikasi Data Rasio *Imbalance* 15 Menggunakan Parameter Optimum

Box dan *whisker* plot digunakan untuk melihat pola penyebaran performansi g-mean kempat metode yang digunakan pada 20 replikasi. Berdasarkan Gambar 4.7 dapat diketahui bahwa SMOTEBoost cenderung menghasilkan nilai-nilai performansi g-mean yang lebih tinggi dengan variasi yang lebih kecil jika dibandingkan dengan ketiga metode lain yang digunakan. Sementara SVM menghasilkan nilai-nilai g-mean yang cenderung kecil dengan variasi yang lebih

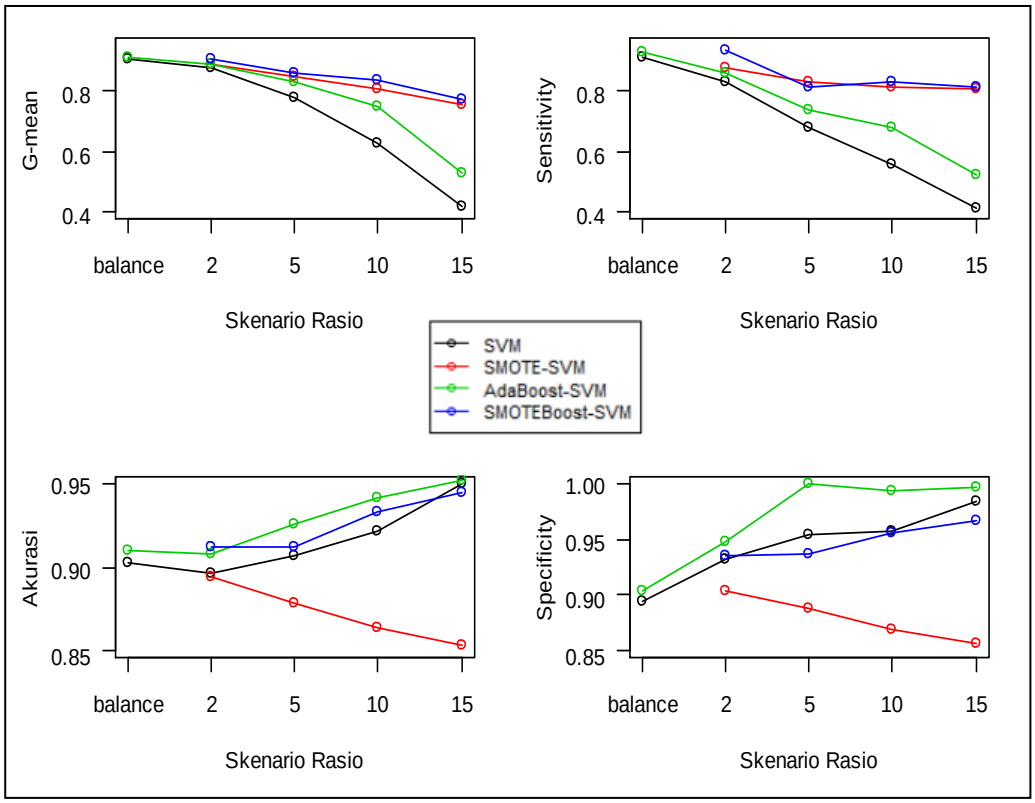
besar dibandingkan ketiga metode lain yang digunakan. *Summary* performansi g-mean untuk keempat metode pada 20 replikasi ditampilkan pada tabel 4.10.

Tabel 4.10 *Summary* Performansi Klasifikasi Data Rasio Imbalance 15 Menggunakan Parameter Optimum

	G-mean					
Metode	Min	Q1	Median	Q 3	Mean	Max
SVM	0	0,1944	0,3679	0,6340	0,4211	1
AdaBoost-SVM	0	0,3835	0,5361	0,5947	0,5262	0,989
SMOTE-SVM	0,3448	0,6492	0,7440	0,8603	0,7377	1
SMOTEBoost-SVM	0,2777	0,7235	0,7700	0,8549	0,7762	0,9947

4.3.7 Perbandingan Performansi Klasifikasi

Pada sub bab ini akan dilihat perbandingan performansi dari SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM dalam pengklasifikasin data bangkitan pada studi simulasi yang dilkukan menggunakan ukuran akurasi, *sensitivity*, *specificity*, dan g-mean.



Gambar 4.8 Perbandingan Performansi Model pada Data Simulasi

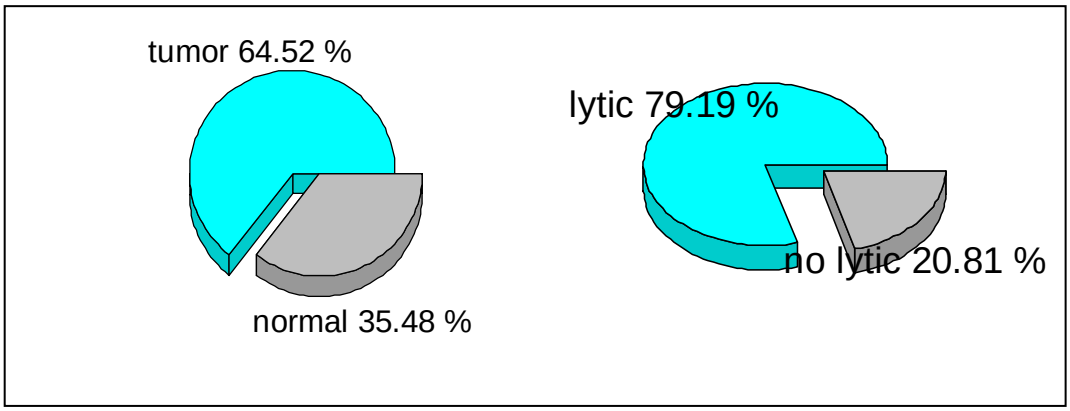
Berdasarkan gambar 4.8 dapat dilihat bahwa secara umum semakin tinggi tingkat rasio *imbalance* pada data maka *classifier* semakin sulit untuk mengklasifikasikan pengamatan yang berasal dari kelas minoritas dengan benar. Hal ini dapat dilihat dengan performansi *sensitivity* pada keempat metode baik SVM, SMOTE-SVM, AdaBoost-SVM, dan SMOTEBoost-SVM yang mengalami penurunan seiring bertambahnya tingkatan rasio *imbalance* pada data. Berdasarkan simulasi ini, dapat diketahui bahwa secara umum SVM cenderung menghasilkan performansi yang palik buruk diantara ketiga metode lainnya, hal ini dapat dilihat melalui Gambar 4.8, dimana SVM cenderung mengalami penurunan performansi g-mean dan *sensitivity* paling drastis seiring bertambahnya tingkat rasio *imbalance*. SVM mengalami penurunan performansi G-mean dan *sensitivity* drastis pada saat melakukan klasifikasi data dengan rasio *imbalance* > 2.

Berdasarkan Gambar 4.8 dapat diketahui pula bahwa AdaBoost-SVM unggul jika dilihat dari nilai akurasi dan *specificity*. Hal ini dikarenakan pada proses *boosting*-nya, AdaBoost-SVM berhasil mengambil keuntungan dari kesalahan-kesalahan klasifikasi yang dilakukan oleh SVM sebagai *base classifier* sehingga pada iterasi-iterasi selanjutnya, *base classifier* dipaksa untuk lebih fokus pada pengamatan-pengamatan yang sulit diklasifikasikan sehingga dapat meningkatkan nilai ketepatan klasifikasi keseluruhan. Namun jika dilihat melalui ukuran g-mean yang mana merupakan ukuran yang lebih diutamakan pada kasus data *imbalance*, AdaBoost-SVM cenderung mengalami penurunan cukup drastis pada rasio *imbalance* > 5 yang berarti bahwa berdasarkan simulasi ini, metode ini kurang baik digunakan pada klasifikasi data *microarray* dengan rasio *imbalance* > 5.

Berdasarkan hasil simulasi ini diketahui bahwa SMOTEBoost-SVM cenderung paling unggul dalam performansi g-mean dan *sensitivity* dibandingkan ketiga metode lain. Lebih jauh lagi, performansi yang dihasilkan juga cenderung stabil dalam mengklasifikasikan data *microarray* dengan beberapa tingkatan rasio *imbalance* yang berbeda. Hal ini dikarenakan kombinasi dari algoritma SMOTE dan prosedur *boosting*-nya sehingga memaksa *base classifier* untuk tidak hanya fokus pada kelas mayoritas saja tetapi juga lebih fokus pada kelas minoritas.

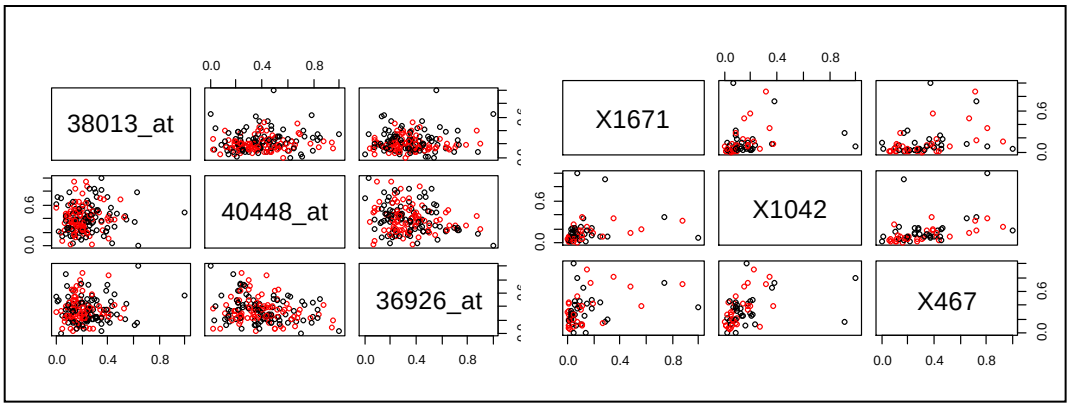
4.4 Klasifikasi Pada Data *Microarray*

Pada bagian ini akan terlebih dahulu disajikan karakteristik dari data kanker colon dan myeloma. Setiap data tersebut memiliki karakteristik berbeda dilihat dari pola persebaran data dari setiap atribut-atribut dan kategori kelasnya. Betikut merupakan karakteristik dari masing-masing data.



Gambar 4.9 Distribusi Kelas pada Masing-Masing Data: (kiri) Kanker Colon; (kanan) Myeloma

Berdasarkan Gambar 4.9 dapat dilihat bahwa distribusi kelas pada kedua data yang digunakan adalah *imbalance* dimana salah satu kelas terdistribusi lebih banyak dibandingkan kelas yang lain. Selanjutnya akan disajikan persebaran data pengamatan dari beberapa pasangan fitur-fitur informatif pada data kanker colon dan myeloma melalui Gambar 4.10.



Gambar 4.10 Pola Persebaran Kelas Pengamatan Data Kanker Colon (kanan) dan Myeloma (kiri) pada Beberapa Fitur Informatif

Berdasarkan gambar 4.10 dapat diketahui bahwa persebaran kelas pengamatan terlihat acak. Pada gambar tersebut terlihat bahwa terdapat indikasi tidak ada

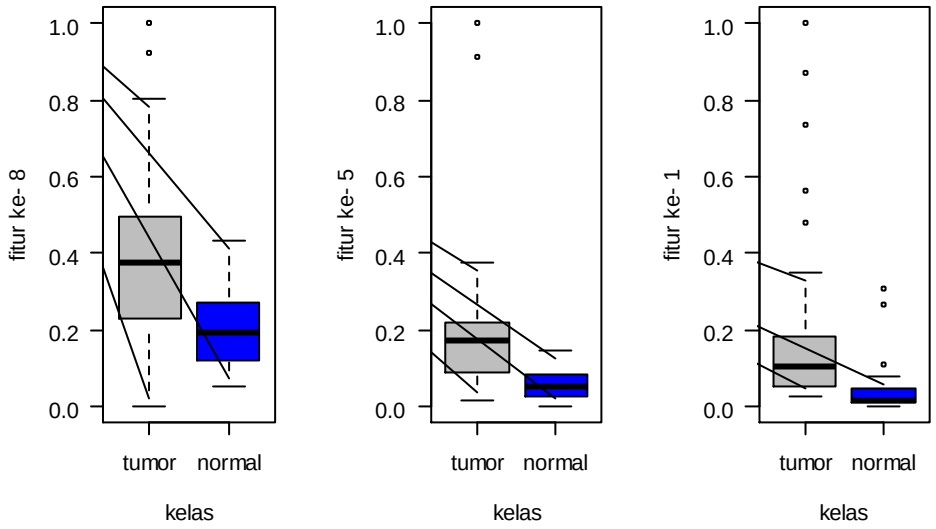
fungsi pemisah linier yang dapat memisahkan kedua kelas pengamatan pada beberapa pasangan fitur informatif pada kedua data yang digunakan. Oleh karena intu dipilih klasifikasi non linier menggunakan SVM.

Sebelum melakukan klasifikasi, pada penelitian ini seleksi fitur dilakukan guna membuang gen-gen yang kurang informatif dan hanya memilih gen-gen yang informatif yang akan digunakan dalam klasifikasi. Hasil seleksi fitur menggunakan metode FCBF pada kedua datasets disajikan pada tabel berikut:

Tabel 4.11 Jumlah Fitur Sebelum dan Setelah Seleksi Fitur

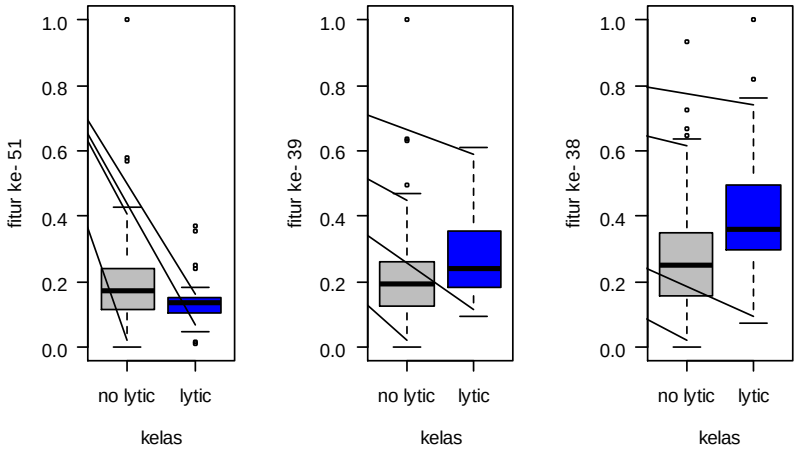
Data	Sebelum Seleksi Fitur	Setelah Seleksi Fitur
Kanker Colon	2000	15
Myeloma	12625	59

Dengan menggunakan *threshold* untuk nilai SU sebesar 0,05 diperoleh hanya 15 gen yang informatif pada data kanker colon dan 59 gen informatif pada data myeloma. Fitur-fitur informatif yang terpilih disajikan dalam lampiran. Berikut boxplot nilai-nilai dari beberapa fitur informatif untuk data kanker colon.



Gambar 4.11 *Boxplot* Nilai-Nilai Ekspresi Gen pada Beberapa Fitur Informatif pada Data Kanker Colon

Berdasarkan Gambar 4.11 dapat dilihat bahwa pada beberapa fitur informatif, nilai-nilai pada kelas normal cenderung lebih kecil dibandingkan nilai-nilai pada kelas tumor.

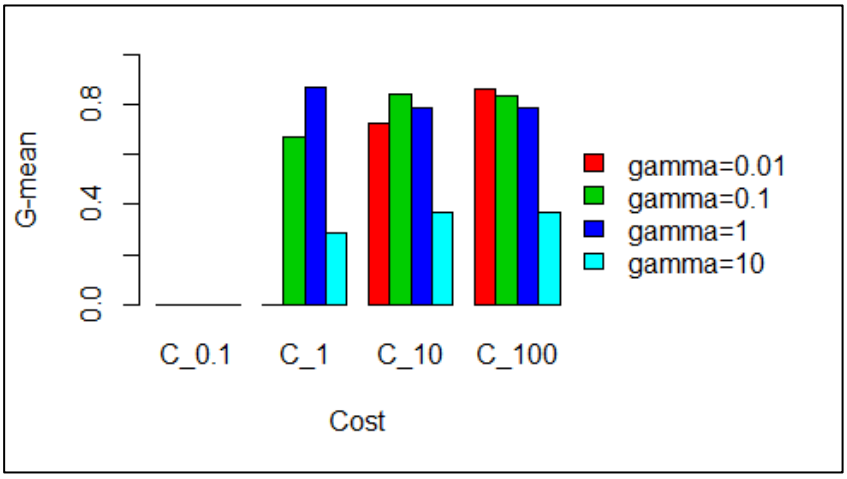


Gambar 4.12 *Boxplot* Nilai-Nilai Ekspresi Gen pada Beberapa Fitur Informatif pada Data Myeloma

Gambar 4.12 menyajikan *boxplot* nilai-nilai dari beberapa fitur informatif pada data myeloma yang terpilih dalam proses seleksi fitur menggunakan FCBF. Berdasarkan Gambar 4.12 dapat dilihat bahwa pada beberapa fitur informatif, nilai-nilai pada kelas tidak terdapat luka (abu-abu) cenderung memiliki pola yang berbeda dibandingkan nilai-nilai pada kelas terdapat luka pada tulang belakang (biru). Selanjutnya akan dilakukan klasifikasi menggunakan SVM dan SMOTE-SVM pada data kanker colon dan myeloma dengan menggunakan seluruh fitur (gen) dan dengan menggunakan fitur yang terpilih guna membandingkan hasilnya. Selanjutnya akan disajikan hasil pengklasifikasian algoritma SVM, SMOTE-SVM, Adaboost-SVM dan SMOTEBoost-SVM pada data kanker colon dan data myeloma. Tahapan pertama yang dilakukan yaitu *scaling* data. Scaling data bertujuan untuk mengatasi permasalahan dimana atribut numerik (gen) dengan *range* data yang besar mendominasi atribut numerik (gen) lainnya yang *range* data nya kecil.

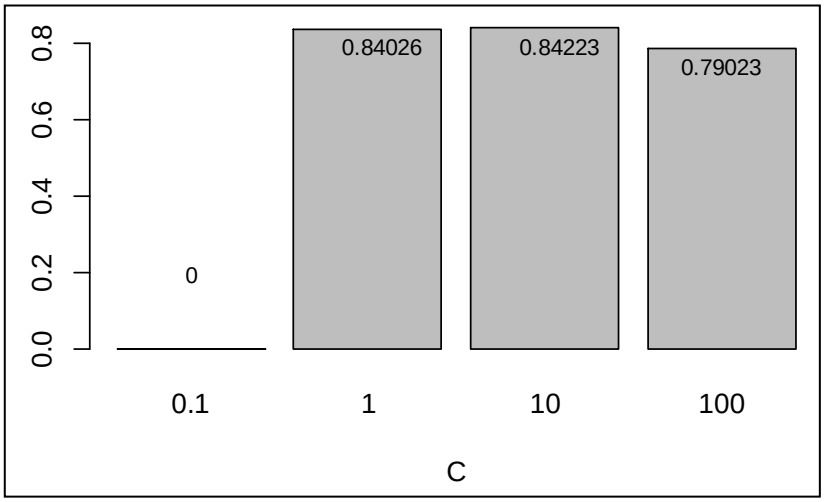
4.4.1 Klasifikasi Menggunakan SVM

Setelah seleksi fitur dilakukan, kemudian klasifikasi dilakukan pada kedua data. Klasifikasi pertama yaitu dengan menggunakan SVM (*Support Vector Machine*). Fungsi kernel yang digunakan yaitu fungsi kernel linier dan radial. Parameter C dan γ dipilih dengan range C= 0,1;1;10;100 dan γ = 0,01;0,1;1;10. Kemudian parameter optimal ditentukan berdasarkan nilai rata-rata *g-mean* tertinggi dari hasil 5-*fold cross validation* dengan stratifikasi. Nilai-nilai g-mean yang dihasilkan dapat dilihat pada pada Lampiran 11.



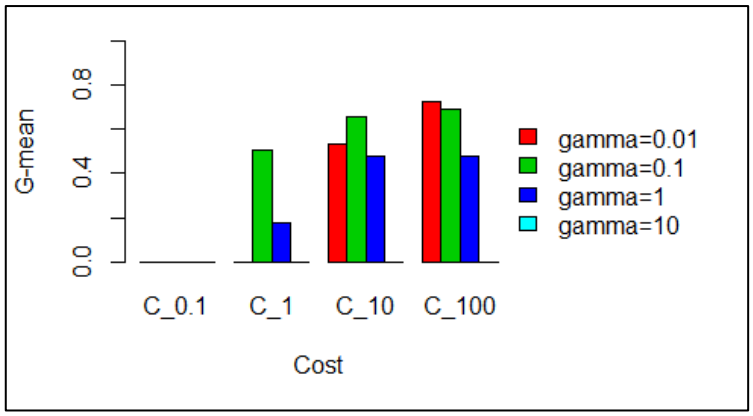
Gambar 4.13 Performansi G-mean pada Klasifikasi SVM pada Data Kanker Colon Menggunakan Beberapa Nilai C dan γ Kernel Radial

Berdasarkan Gambar 4.13 dapat dilihat bahwa klasifikasi dengan SVM pada data kanker colon menggunakan kernel radial memberikan nilai g-mean terbesar yaitu 0,87 pada pasangan parameter C=1 dan γ =1. dapat dilihat pula bahwa pada nilai C=0,1, SVM menghasilkan nilai g-mean yang sama untuk setiap nilai γ yang digunakan. Secara umum, nilai g-mean cenderung mengalami peningkatan seiring peningkatan nilai C pada γ=0,01, 0,1, dan 10 tapi cenderung tidak pada γ=1.



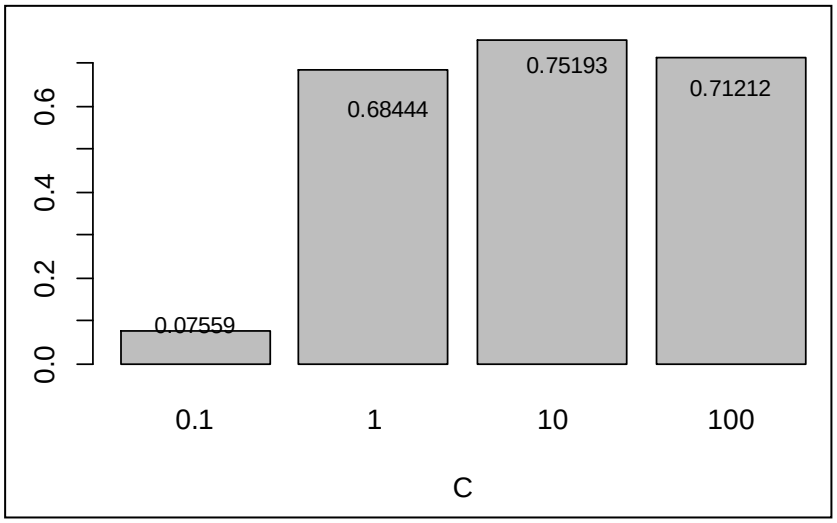
Gambar 4.14 Performansi G-mean pada Klasifikasi SVM pada Data Kanker Colon Menggunakan Beberapa Nilai C Pada Kernel Linier

Dengan menggunakan kernel linier, nilai g-mean terbesar yaitu 0,8422 pada C=10. Sehingga parameter optimum SVM untuk data kanker kolon pada selang parameter yang telah ditentukan dengan menggunakan kernel radial dan linier diatas yaitu C=1 dan γ=1 dengan nilai g-mean sebesar 0,87. Selanjutnya, dilakukan klasifikasi data myeloma dengan *setting* percobaan yang sama dengan data kanker colon. Gambar 4.15 menyajikan nilai g-mean pada klasifikasi data myeloma menggunakan SVM dengan kernel radial.



Gambar 4.15 Performansi G-mean pada Klasifikasi SVM pada Data Myeloma Menggunakan Beberapa Nilai C dan γ pada Kernel Radial

Berdasarkan Gambar 4.15 dapat dilihat bahwa secara umum klasifikasi dengan svm pada data myeloma menggunakan kernel radial cenderung mengalami peningkatan performa g-mean seiring bertambahnya nilai C yang digunakan. Parameter optimum yang menghasilkan nilai g-mean terbesar yaitu pasangan C=100 dan γ=0,01 dengan nilai g-mean sebesar 0,7234.



Gambar 4.16 Performansi G-mean pada Klasifikasi SVM pada Data Myeloma Menggunakan Beberapa Nilai C Kernel Linier

Nilai g-mean terbesar yang dihasilkan dalam klasifikasi data myeloma menggunakan kernel linier yaitu dengan parameter cost C=10 dengan nilai g-mean sebesar 0,7519. Sehingga pada klasifikasi data myeloma, model optimum dengan menggunakan kedua fungsi kernel yaitu dengan menggunakan kernel linier dengan parameter cost C=10 dengan nilai g-mean sebesar 0,7519. Rangkuman rata-rata performansi dari 5-*fold cross validation* pada parameter optimum pada kedua *dataset* dengan menggunakan seluruh fitur dan fitur-fitur terpilih yaitu sebagai berikut

Tabel 4.12 Rangkuman Performansi pada Klasifikasi SVM Menggunakan Seluruh Fitur dan Beberapa Fitur Informatif

Data	Fitur	Model SVM optimum	Rata-Rata			
			Akurasi	*Specificity*	*Sensitivity*	*G-mean*
Kanker Colon (RI= 1,82)	Seluruh Fitur	Radial C=100;γ= 0,01	0,8218	0,9	0,69	0,7743
		Linier C=0,1	0,8218	0,875	0,74	0,7926
	Fitur Terpilih	Radial C=1; γ =1	**0,8872**	**0,925**	**0,83**	**0,87**
		Rinier C=10	**0,8538**	**0,875**	**0,83**	**0,8422**
Myeloma (RI=3,81)	Seluruh Fitur	Radial C=10; γ=0,01	0,7919	**1**	0	0
		Linier C=10	0,8094	**0,9638**	0,2143	0,3935
	Fitur Terpilih	Radial C=100; γ= 0,01	**0,855**	0,9267	**0,5821**	**0,7234**
		Kernel linier C=10	**0,855**	0,9122	**0,6357**	**0,7519**

*RI=Rasio *Imbalance*

Berdasarkan Tabel 4.12, dapat dilihat bahwa dengan melakukan seleksi fitur menggunakan FCBF sebelum melakukan klasifikasi menggunakan SVM dapat meningkatkan performansi nilai g-mean untuk kedua data baik pada data kanker colon maupun myeloma. Dapat dilihat pula bahwa SVM mengalami penurunan performansi khususnya g-mean pada klasifikasi data myeloma dibandingkan data kanker colon, hal ini dikarenakan data myeloma memiliki rasio *imbalance* lebih besar dibandingkan (>2 kali) data kanker colon sehingga membuat SVM lebih sulit dalam mengklasifikasikan data tersebut. Berdasarkan tabel diatas juga dapat diketahui bahwa penggunakan seluruh fitur pada klasifikasi kedua data diperoleh hasil bahwa kernel linier lebih baik dibandingkan menggunakan kernel radial, hal ini sesuai dalam penelitian Hsu dkk (2003) yang menyatakan bahwa jika jumlah fitur > jumlah pengamatan maka direkomendasikan untuk menggunakan kernel linier dalam klasifikasi menggunakan SVM. Pada Tabel 4.12 juga terlihat performansi SVM kurang baik dalam mengklasifikasikan kelas minoritas untuk kedua data. Jika dilihat dari nilai-nilai *sensitivity* yang diperoleh lebih kecil dibanding kan nilai-nilai *specificity* nya. Hal ini terjadi karena pada data yang *imbalance* fungsi pemisah SVM cenderung lebih mudah mengklasifikasikan data ke dalam kelas mayor (negatif). Selanjutnya, algoritma SMOTE dilakukan untuk menyeimbangkan distribusi kelas dengan menambah sampel dari pengamatan kelas minor. SMOTE berfokus untuk meningkatkan ketepatan klasifikasi pada kelas minoritas (*sensitivity*).

4.4.2 Klasifikasi Menggunakan SMOTE-SVM

Pada klasifikasi menggunakan SMOTE-SVM maka data minor pada setiap data *training* di tiap *cross validation* diperbanyak sedemikian hingga distribusi data mayor (negatif) dan minor (positif) cukup seimbang. Sebelumnya dicari terlebih dahulu k-tetangga terdekat dari data yang akan direplikasi menggunakan jarak euclidean. Dalam penelitian ini digunakan 5 tetangga terdekat yang kemudian nantinya akan dipilih secara acak 1 dari 5 pengamatan tersebut untuk menghasilkan sebuah data sintetis. Berikut distribusi kelas dari kedua data sebelum dan setelah dilakukan penambahan data sintetis menggunakan SMOTE pada data setiap *training*.

Tabel 4.13 Distribusi Kelas Pengamatan Sebelum dan Setelah Dilakukan Replikasi pada Kelas Minor

Data	SMOTE	Kelas	Data *training*				
			1	2	3	4	5
Kanker Colon	Sebelum SMOTE	Negatif	64%	64%	64%	65,31%	65,31%
		Positif	36%	36%	36%	34,69%	34,69%
	Setelah SMOTE	Negatif	47,06%	47,06%	47,06%	48,48%	48,48%
		Positif	52,94%	52,94%	52,94%	51,52%	51,52%
Myeloma	Sebelum SMOTE	Negatif	78,99%	79,14%	79,14%	78,99%	79,71%
		Positif	21,01%	20,86%	20,86%	21,01%	20,29%
	Setelah SMOTE	Negatif	48,44%	48,67%	48,67%	48,44%	49,55%
		Positif	51,56%	51,33%	51,53%	51,56%	50,45%

Berdasarkan tabel diatas dapat dilihat bahwa distribusi kelas dari setiap data *training* hampir seimbang setelah dilakukan replikasi pengamatan pada kelas minor sebanyak satu kali pada data kanker colon dan tiga kali pada data myeloma.. Selanjutnya dilakukan klasifikasi menggunakan SVM pada setiap data *trainning* yang telah seimbang. Pada bagian ini proses klasifikasi hanya menggunakan fitur-fitur yang informatif saja dikarenakan telah ditunjukan pada bagian sebelumnya bahwa menggunakan fitur-fitur informatif memberikan hasil yang lebih baik dibandingkan menggunakan seluruh fitur. Rata-rata performansi klasifikasi menggunakan SMOTE-SVM dengan menggunakan kernel radial dan linier pada data kanker colon dengan 5-*fold cross validation* di sajikan pada Tabel 4.14. Berdasarkan Tabel 4.14 dapat dilihat bahwa nilai-nilai g-mean yang

dihasilkan cenderung stabil untuk C=1,10, dan 100 pada klasifikasi menggunakan kernel radial. Adapun nilai g-mean terbesar yakni sebesar 0,8894 dihasilkan dengan menggunakan pasangan parameter C=0,1 dan γ=1. Berdasarkan Tabel 4.14 dapat dilihat pula bahwa pada kernel linier, performansi g-mean yang dihasilkan cenderung menurun seiring bertambahnya nilai parameter C.

Tabel 4.14 Rangkuman Performansi Klasifikasi Data Kanker Colon Menggunakan SMOTE-SVM pada Beberapa Parameter

Kernel	C	γ	Akurasi	*Specificity*	*Sensitivity*	G-mean
Radial	0,1	0,01	0,353846	0	**1**	0
	0,1	0,1	0,353846	0	**1**	0
	0,1	1	0,8667	0,8	**1**	**0,8894**
	0,1	10	0,353846	0	**1**	0
	1	0,01	0,353846	0	**1**	0
	1	0,1	0,8526	0,85	0,87	0,853
	1	1	**0,8705**	0,9	0,83	0,8561
	1	10	0,6949	**0,95**	0,24	0,4179
	10	0,01	0,8692	0,85	0,92	0,8762
	10	0,1	0,8692	0,875	0,87	0,8662
	10	1	0,8231	0,875	0,75	0,7898
	10	10	0,6782	0,925	0,24	0,411
	100	0,01	0,8692	0,875	0,87	0,8662
	100	0,1	0,8551	0,85	0,88	0,8573
	100	1	0,8397	0,9	0,75	0,801
	100	10	0,6782	0,925	0,24	0,411
Llinier	0,1	-	0,8179	0,725	**1**	**0,8447**
	1	-	**0,8538**	**0,875**	0,83	0,8422
	10	-	0,8371	0,85	0,83	0,8302
	100	-	0,8231	**0,875**	0,74	0,7902

Pada klasifikasi data myeloma diperoleh model optimum yaitu dengan menggunakan kernel linier dengan parameter C=1 yang menghasilkan nilai g-mean sebesar 0,8141.

Tabel 4.15 Rangkuman Performansi Klasifikasi Data Myeloma Menggunakan SMOTE-SVM pada Beberapa Parameter

Kernel	C	γ	Akurasi	*Specificity*	*Sensitivity*	G-mean
Radial	0,1	0,01	0,2081	0,0000	**1,0000**	0,0000
	0,1	0,1	0,7403	0,7019	0,8929	0,7877
	0,1	1	0,7919	0,9132	0,3357	0,4683
	0,1	10	0,2081	0,0000	**1,0000**	0,0000
	1	0,01	0,7928	0,7815	0,8143	0,7928
	1	0,1	0,8264	0,8466	0,7571	0,7947
	1	1	0,8150	0,9489	0,3036	0,5285
	1	10	0,7919	**1,0000**	0,0000	0,0000
	10	0,01	0,8323	0,8537	0,7571	**0,7991**
	10	0,1	**0,8726**	0,9415	0,6143	0,7505
	10	1	0,8091	0,9489	0,2750	0,4978
	10	10	0,7919	**1,0000**	0,0000	0,0000
	100	0,01	0,8437	0,9124	0,5857	0,7234
	100	0,1	0,8380	0,9196	0,5286	0,6878
	100	1	0,8150	0,9563	0,2750	0,4992
	100	10	0,7919	**1,0000**	0,0000	0,0000
Linier	0,1	-	0,8326	0,8542	**0,7571**	0,7992
	1	-	**0,8842**	**0,9270**	0,7250	**0,8141**
	10	-	0,8435	0,9050	0,6071	0,7316
	100	-	0,8378	0,9050	0,5786	0,7121

Rangkuman rata-rata performansi dari 5-*fold cross validation* pada parameter optimum pada kedua *dataset* disajikan pada tabel berikut:

Tabel 4.16 Rangkuman Performansi Klasifikasi SMOTE-SVM pada Parameter Optimum

Data	Model SMOTE-SVM	Rata-Rata			
		Akurasi	*Sensitivity*	*Specificity*	G-mean
Kanker Colon	Radial C=0.1; γ=1	0,8667	1	0,8	0,8894
	Linier C=0,1	0,8179	1	0,725	0,8447
Myeloma	Radial C=10; γ=0,01	0,8323	0,7571	0,8537	0,7991
	Linier C=1	0,8842	0,725	0,927	0,8141

Pada hasil klasifikasi menggunakan SVM pada data yang telah diseimbangkan dengan SMOTE diperoleh peningkatan nilai *sensitivity* atau ketepatan klasifikasi pada kelas minor (positif) jika dibandingkan klasifikasi pada data tanpa SMOTE.

4.4.3 Klasifikasi Menggunakan *Boosting*-SVM

Pada bagian ini diperlihatkan peenggunaan *boosting* dengan *base classifier* SVM untuk kedua *dataset*. Metode *boosting* yang digunakan yaitu AdaBoost dan SMOTEBoost. Perbedaan kedua metode *boosting* tersebut terletak pada penggunaan SMOTE di tiap iterasi *boosting*. SMOTE dalam SMOTEBoost bertujuan memodifikasi distribusi D_t sehingga menambah probabilitas untuk kelas minor terpilih sebagai sampel ke dalam *training set* di setiap iterasi *boosting* yang mana pada metode AdaBoost didominasi oleh kelas mayoritas. Iterasi maksimal yang digunakan yakni berubah-ubah dari hingga maksimal 20 iterasi, yang mana akan diperoleh iterasi optimum yang menghasilkan nilai g-mean terbesar. Selain itu iterasi akan berhenti pula ketika nilai *trainning error* pada suatu iterasi bernilai 0. Pada kasus seperti ini metode *boosting* tidak dapat mengambil keuntungan dari *base classifier*. *Setting* parameter SVM dan fungsi kernel yang digunakan sama seperti pada sub bab sebelumnya. Sub bab selanjutnya akan menyajikan hasil klasifikasi menggunakan AdaBoost-SVM.

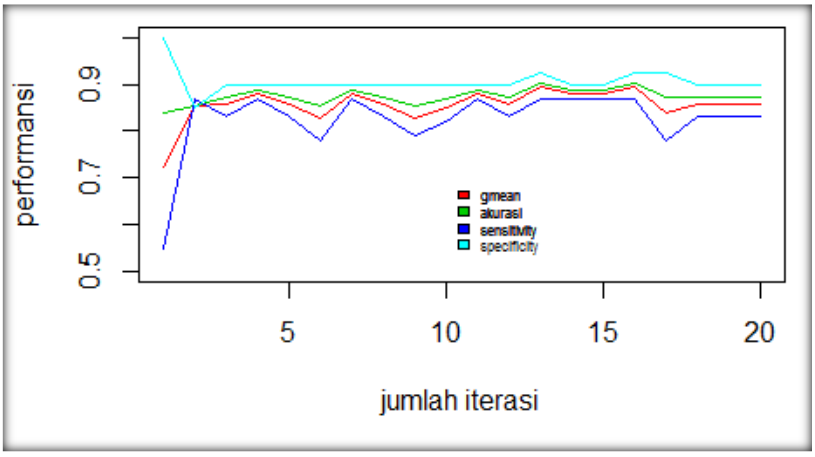
4.4.3.1 Klasifikasi Menggunakan AdaBoost-SVM

Hasil performansi rata-rata g-mean pada kelima *fold* menggunakan boosting *AdaBoost*-SVM pada data kanker colon dan myeloma dengan menggunakan semua *range* parameter yang dicobakan ditampilkan sebagai pada Tabel 4.17. Pada Tabel 4.17 dapat dilihat bahwa klasifikasi menggunakan beberapa parameter tidak dapat menghasilkan nilai g-mean dikarenakan AdaBoost tidak dapat mengambil keuntungan dari *base classifier* tersebut. Hal tersebut terjadi karena nilai *error* klasifikasi pada *training set* di iterasi pertama mengsasilkan nilai 0, sehingga iterasi tidak dapat dilanjutkan.

Tabel 4.17 Rangkuman Performansi G-mean Pada Klasifikasi Kedua Data Menggunakan AdaBoost-SVM dengan Beberapa Parameter

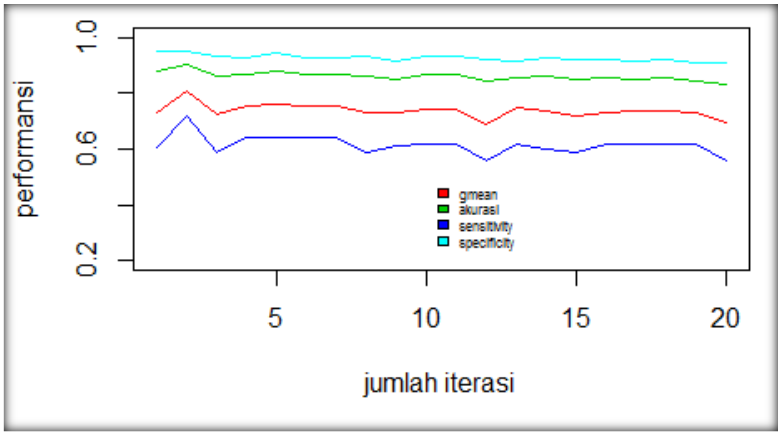
Data							
Kanker Colon				Myeloma			
Kernel	C	γ	G-mean	Kernel	C	γ	G-mean
Radial	0,1	0,01	0,0000	Radial	0,1	0,01	0,0000
	0,1	0,1	0,0000		0,1	0,1	0,1050
	0,1	1	0,6385		0,1	1	0,0000
	0,1	10	0,0000		0,1	10	0,0000
	1	0,01	0,0000		1	0,01	0,3978
	1	0,1	0,8901		1	0,1	0,7845
	1	1	0,8700		1	1	-
	1	10	-		1	10	-
	10	0,01	**0,8930**		10	0,01	0,7562
	10	0,1	0,8913		10	0,1	**0,8102**
	10	1	-		10	1	-
	10	10	-		10	10	-
	100	0,01	0,8662		100	0,01	0,7802
	100	0,1	-		100	0,1	0,0000
	100	1	-		100	1	-
	100	10	-		100	10	-
Linier	0,1	-	**0,8829**	Linier	0,1	-	**0,7869**
	1	-	0,8673		1	-	0,7696
	10	-	0,8690		10	-	-
	100	-	-		100	-	-

Berdasarkan Tabel 4.17 diperoleh nilai g-mean terbesar pada klasifikasi data kanker colon yakni 0,8930 diperoleh saat menggunakan kernel radial dengan parameter C=10 dan γ=0,01 sedangkan pada klasifikasi data myeloma diperoleh nilai g-mean terbesar 0,8102 saat menggunakan kernel radial dengan parameter C=10 dan γ=0,1



Gambar 4.17 Plot Nilai Performansi pada Beberapa Iterasi Maksimal AdaBoost-SVM pada Klasifikasi Data Kanker Colon

Gambar 4.17 menyajikan performansi klasifikasi data kanker kolon pada parameter optimum. Berdasarkan gambar diatas dapat dilihat bahwa nlai akurasi, *sensitivity*, *specificity*, dan g-mean cenderung mencapai maksimal pada iterasi ke 13. Pada klasifikasi data myeloma, jumlah iterasi yang menghasilkan performansi g-gmean, akurasi, *sensitivity*, dan *specificity* tertinggi yaitu 2 iterasi. Kurva performansi untuk masing-masing jumlah iterasi yang digunakan ditunjukan pada Gambar 4.18



Gambar 4.18 Plot Nilai Performansi pada Beberapa Iterasi Maksimal AdaBoost-SVM pada Klasifikasi Data Myeloma

Rangkuman performansi hasil klasifikasi dengan parameter optimum ditampilkan pada tabel berikut:

Tabel 4.18 Rangkuman Performansi Klasifikasi AdaBoost-SVM pada Parameter Optimum

Data	Model Optimum	Rata-Rata			
		Akurasi	*Sensitivity*	*Specificity*	G-mean
Kanker Colon	Radial C=10; γ=0,01; 13 iterasi	0,9026	0,87	0,925	0,893
	Linier C=0,1; 18 iterasi	0,9038	0,83	0,95	0,8829
Myeloma	Radial C=10; γ=0,1; 2 iterasi	0,9017	0,7179	0,9492	0,8102
	Linier C=0,1;19 iterasi	0,8901	0,6464	0,9269	0,7869

Nilai akurasi yang diperoleh pada pengklasifikasian kanker colon yakni 0,9026 pada nilai g-mean terbesar, yang berarti 90,38 persen pengamatan diklasifikasikan dengan benar pada saat menggunakan kernel linier. Nilai *sensitivity* yang di peroleh sebesar 0,87 yang berarti fungsi pemisah yang diperoleh mampu mendeteksi 87% pengamatan yang berasal dari kelas normal dengan benar. Sementara nilai *specificity* sebesar 0,925 berarti fungsi pemisah yang diperoleh berhasil mengidentifikasi 92,5% pengamatan yang berasal dari kelas tumor.

Pada data myeloma diperolah akurasi sebesar 0,9017 yang berarti 90,17% pengamatan dapat diklasifikasikan dengan benar pada saat menggunakan kernel radial. Nilai *sensitivity* yang di peroleh sebesar 0,7179 yang berarti fungsi pemisah yang diperoleh mampu mendeteksi 71,79%% pengamatan yang berasal dari kelas tidak ada luka dengan benar. Sementara nilai *specificity* sebesar 0,9492 berarti fungsi pemisah yang diperoleh berhasil mengidentifikasi 94,92% pengamatan yang berasal dari kelas terdapat luka pada tulang belakang.

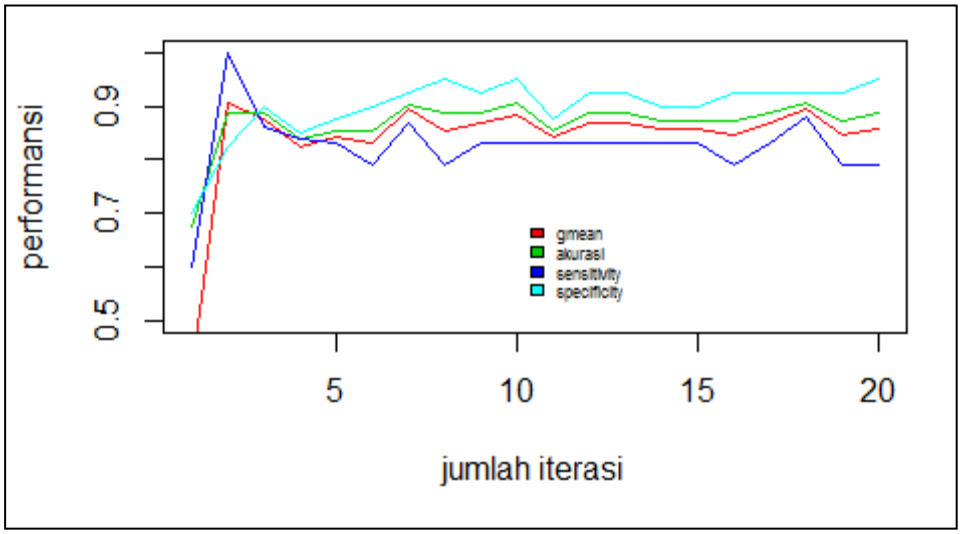
4.4.3.2 Klasifikasi Menggunakan SMOTEBoost-SVM

Pada kedua dataset dilakukan replikasi pada kelas minor smasing-masing sebanyak 1 pada data kanker colon dan 3 pada data myeloma pada tiap iterasi *boosting*. Hasil performansi rata-rata g-mean pada kelima *fold* menggunakan *boosting* SMOTEBoost-SVM pada data kanker colon dan myeloma dengan menggunakan semua *range* parameter yang dicobakan ditampilkan sebagai berikut.

Tabel 4.19 Rangkuman Performansi G-mean Menggunakan SMOTEBoost-SVM Dengan Beberapa Parameter

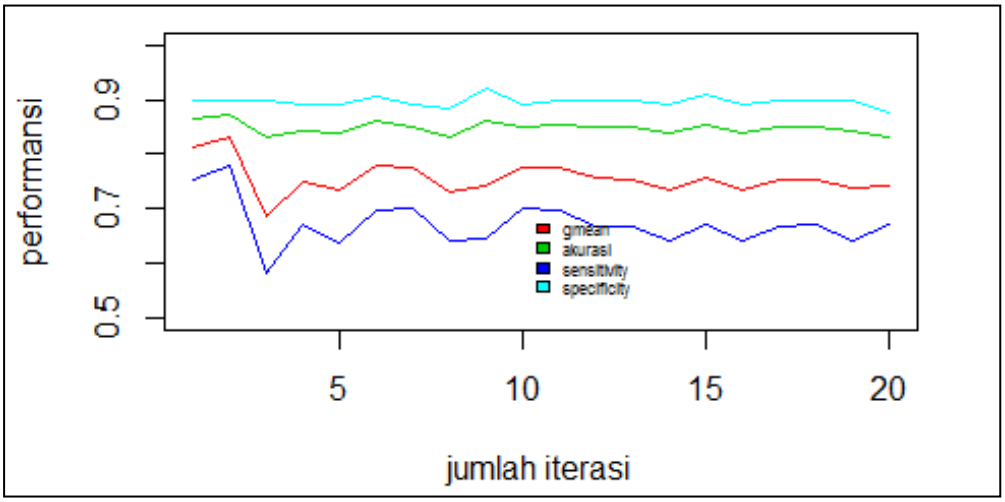
Data									
Kanker Colon					Myeloma				
Kernel	C	γ	G-mean	Iterasi optimum	Kernel	C	γ	G-mean	Iterasi optimum
Radial	0,1	0,01	0,1414	11	Radial	0,1	0,01	0,1361	1
	0,1	0,1	0,0000	-		0,1	0,1	0,7726	1
	0,1	1	**0,9055**	2		0,1	1	0,4502	4
	0,1	10	0,1732	9		0,1	10	0,0000	-
	1	0,01	0,1414	16		1	0,01	0,8019	1
	1	0,1	0,8819	8		1	0,1	**0,8294**	3
	1	1	0,8410	2		1	1	0,6481	17
	1	10	0,7296	9		1	10	0,0000	-
	10	0,01	0,8801	17		10	0,01	0,8268	2
	10	0,1	0,8873	1		10	0,1	0,7564	1
	10	1	0,8295	14		10	1	0,6727	8
	10	10	0,6611	13		10	10	0,0000	-
	100	0,01	0,8873	1		100	0,01	0,7987	2
	100	0,1	0,8841	14		100	0,1	0,7728	2
	100	1	0,8032	3		100	1	0,6593	11
	100	10	0,7266	3		100	10	0,1134	9
Linier	0,1	-	0,8940	14	Linier	0,1	-	0,8254	13
	1	-	0,8913	5		1	-	**0,8320**	2
	10	-	**0,9051**	2		10	-	0,7879	2
	100	-	0,8651	5		100	-	0,7977	3

Berdasarkan Tabel 4.18 diperoleh model dengan kernel radial menggunakan parameter C=0,1 dan γ=1 menghasilkan nilai g-mean terbesar pada klasifikasi data kanker colon yakni 0,8930 sedangkan pada klasifikasi data myeloma diperoleh nilai g-mean terbesar 0,8102 saat menggunakan kernel linier dengan parameter C=1.



Gambar 4.19 Plot Nilai Performansi pada Beberapa Iterasi Maksimal SMOTEBoost-SVM pada Klasifikasi Data Kanker Colon

Gambar 4.19 menyajikan performansi klasifikasi data kanker kolon pada tiap iterasi yang digunakan dengan menggunakan parameter optimum. Berdasarkan gambar di atas dapat dilihat bahwa nlai akurasi, *sensitivity*, *specificity*, dan g-mean cenderung mencapai maksimal pada iterasi ke 2. Pada klasifikasi data myeloma, jumlah iterasi yang menghasilkan performansi g-mean, akurasi, *sensitivity*, dan *specificity* tertinggi yaitu 2 iterasi. Kurva performansi untuk masing-masing jumlah iterasi yang digunakan ditunjukan pada Gambar 4.20.



Gambar 4.20 Plot Nilai Performansi pada Beberapa Iterasi Maksimal SMOTEBoost-SVM pada Klasifikasi Data Myeloma

Rangkuman performansi hasil klasifikasi dengan parameter optimum ditampilkan pada tabel berikut:

Tabel 4.20 Rangkuman Performansi Klasifikasi SMOTEBoost-SVM pada Parameter Optimum

Data	Model	Rata-Rata			
		Akurasi	*Sensitivity*	*Specificity*	G-mean
Kanker Colon	Radial C=0,1; γ=1; 2 iterasi	0,8859	1	0,825	0,9055
	Linier C=10; 2 iterasi	0,9026	0,92	0,9	0,9051
Myeloma	Radial C=1; γ=0,1; 3 iterasi	0,8669	0,7786	0,8905	0,8294
	Linier C=1; 2 iterasi	0,8726	0,7786	0,8979	0,832

Nilai akurasi yang diperoleh pada pengklasifikasian data kanker colon yakni 0,8859 pada nilai g-mean terbesar, yang berarti 88,59 persen pengamatan diklasifikasikan dengan benar pada saat menggunakan kernel radial. Nilai *sensitivity* yang di peroleh sebesar 1 yang berarti fungsi pemisah yang diperoleh mampu mendeteksi 100% pengamatan yang berasal dari kelas normal dengan benar. Sementara nilai *specificity* sebesar 0,825 berarti fungsi pemisah yang diperoleh berhasil mengidentifikasi 82,5% pengamatan yang berasal dari kelas tumor dengan benar.

Pada data myeloma diperolah akurasi sebesar 0,8726 yang berarti 87,26% pengamatan diklasifikasikan dengan benar pada saat menggunakan kernel linier. Nilai *sensitivity* yang di peroleh sebesar 0,7786 yang berarti fungsi pemisah yang diperoleh mampu mendeteksi 77,86%% pengamatan yang berasal dari kelas tidak ada luka dengan benar. Sementara nilai *specificity* sebesar 0,8979 berarti fungsi pemisah yang diperoleh berhasil mengidentifikasi 89,79% pengamatan yang berasal dari kelas terdapat luka pada tulang belakang.

4.4.4 Perbandingan Performansi Klasifikasi

Setelah dilakukan analisis pada data kanker colon dan myeloma menggunakan SVM, SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM, selanjunya pada bagian ini dilakukan perbandingan dari performansi semua model optimum yang diperoleh. Perbandingan beberapa metode tersebut diukur menggunakan performansi klasifikasi yang meliputi g-mean, akurasi, *sensitivity*, dan *specificity* yang merupakan hasil klasifikasi dengan parameter terbaik dari

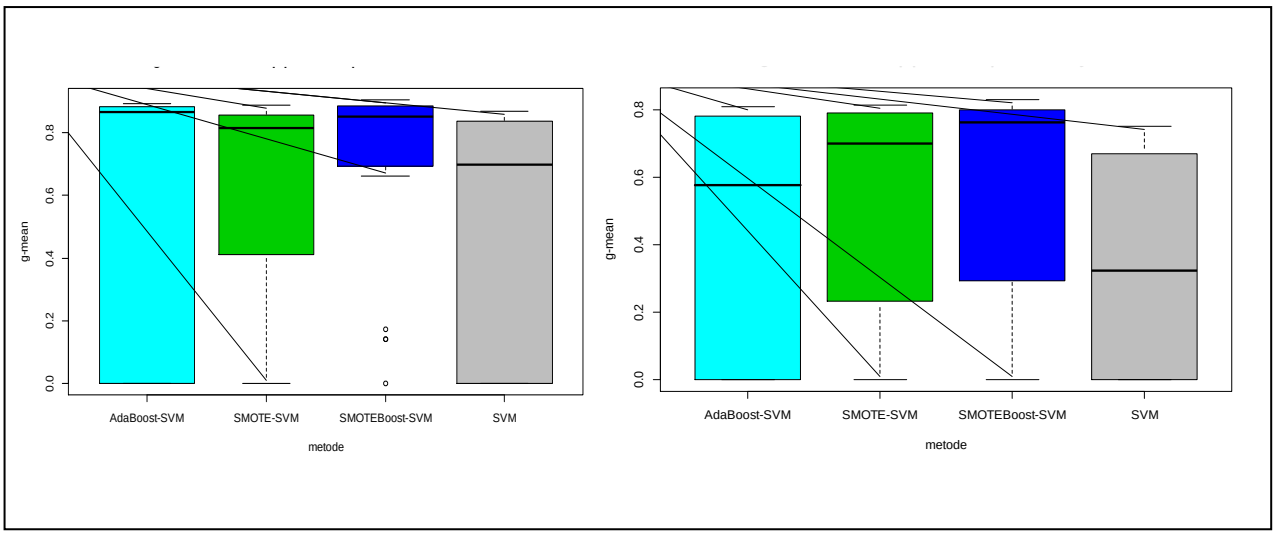
masing-masing metode. Perbandingan hasil klasifikasi tersebut ditampilkan pada Tabel 4.21. Berdasarkan Tabel 4.21, dapat dilihat bahwa performansi dari seluruh model khususnya dilihat dari nilai g-mean mengalami penurunan pada data myeloma dibandingkan dengan data kanker colon. Hal ini dikarenakan rasio *imbalance* dari data myeloma cenderung 2 kali lebih besar dibandingkan data colon sehingga pengamatan-pengamatan pada kelas minoritas pada data myeloma lebih sulit diklasifikasikan dibandingkan pada data kanker colon. Ketepatan klasifikasi yang dilakukan oleh model SVM yaitu rata-rata dari klasifikasi kelima *fold* sebesar 88,72% pada data kanker colon yang berarti rata-rata 88,72% dari pengamatan di tiap fold data colon telah diklasifikasikan dengan benar.

Jika dilihat dari nilai *sensitivity* dan *specificity*, SVM berhasil mengklasifikasikan 83% pengamatan yang berasal dari kelas normal (minoritas) sebagai kelas normal dan berhasil mengklasifikasikan pengamatan 92,5% pengamatan yang berasal dari kelas tumor (mayoritas) sebagai kelas tumor. Berdasarkan hasil tersebut maka SVM lebih baik dalam mengklasifikasikan pengataman kelas tumor dibandingkan kelas normal pada data colon. Lebih jauh lagi pada data myeloma yang notabene memiliki rasio *imbalance* lebih tinggi, hanya 63,57% pengamatan dari kelas tidak ada luka pada tulang (minoritas) yang tepat diklasifikasikan masuk ke dalam kelas tidak ada luka sementara 91,22% pengamatan dari kelas terdapat luka pada tulang (mayoritas) tepat diklasifikasikan masuk ke dalam kelas terdapat luka pada tulang. Adanya kasus imbalanced pada kedua data menyebabkan rendahnya nilai *sensitivity* dikarenakan fungsi pemisah SVM cenderung mengklasifikan pengamatan ke dalam kelas mayoritas. Setelah dilakukan penyeimbangan data pada kedua kelas dan dilakukan *boosting* diperoleh hasil yang lebih baik. Hal ini dibuktikan dengan performansi g-mean yang diperoleh menggunakan SMOTE-SVM, AdaBoost-SVM dan SMOTEBoost-SVM lebih tinggi dibandingkan SVM pada kedua data.

Tabel 4.21 Perbandingan Performansi Klasifikasi Data *Microarray*

Data	Model	Rata-Rata			
		Akurasi	*Sensitivity*	*Specificity*	G-mean
Kanker Colon	SVM	0,8872	0,83	**0,925**	0,87
	SMOTE-SVM	0,8667	**1**	0,8	0,8894
	AdaBoost-SVM	**0,9026**	0,87	**0,925**	0,893
	SMOTEBoost-SVM	0,8859	**1**	0,825	**0,9055**
Myeloma	SVM	0,855	0,6357	0,9122	0,7519
	SMOTE-SVM	0,8842	0,725	0,9122	0,8141
	AdaBoost-SVM	**0,9017**	0,7179	**0,9492**	0,8102
	SMOTEBoost-SVM	0,8726	**0,7786**	0,8979	**0,832**

Berdasarkan Tabel 4.21 dapat dilihat pula bahwa AdaBoost-SVM menghasilkan nilai akurasi dan *specificity* tertinggi pada kedua data. Hal ini dikarenakan pada proses *boosting*-nya, AdaBoost berhasil mengambil keuntungan dari kesalahan klasifikasi yang dilakukan SVM di tiap iterasi *boosting* sehingga dapat meningkatkan ketepatan klasifikasi khususnya klasifikasi pada kelas mayoritas. Sementara SMOTEBoost menghasilkan nilai *sensitivity* dan g-mean tertinggi pada kedua data dikarenakan proses penyeimbangan distribusi kelas *training set* di tiap iterasi *boosting* sehingga mengakibatkan peningkatan ketepatan klasifikasi pada kelas minoritas.



Gambar 4.21 *Boxplot* Nilai-Nilai Performansi G-mean pada Beberapa Parameter yang Digunakan

Gambar 4.21 menyajikan *boxplot* dari nilai-nilai g-mean yang dihasilkan pada setiap model menggunakan seluruh parameter yang dicobakan. Berdasarkan Gambar 4.21, nilai-nilai g-mean yang dihasilkan menggunakan SMOTEBoost-SVM yang ditunjukkan oleh warna biru pada gambar, cenderung lebih tinggi dari pada metode lain pada kedua data. Lebih jauh lagi, variasi dari g-mean yang dihasilkan oleh algoritma SMOTEBoost-SVM cenderung lebih kecil dibandingkan ketiga algoritma lain pada kedua data. Hal ini mengindikasikan bahwa performansi g-mean yang dihasilkan oleh SMOTEBoost-SVM pada semua parameter lebih stabil dibandingkan ketiga algoritma lainnya pada kedua data.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis yang telah dilakukan, dapat disimpulkan sebai berikut.

1. Pada studi simulasi yang dilakukan dengan skenario pembangkitan data dengan beberapa tingkat rasio *imbalance*, diperoleh hasil bahwa secara umum metode *boosting* SMOTEBoost-SVM unggul dalam performansi g-mean dan *sensitivity* pada klasifikasi data dengan rasio *imbalance* 2, 5, 10, dan 15 dibandingkan ketiga metode lain yakni SVM, SMOTE-SVM, dan AdaBoost-SVM. Variasi nilai-nilai performansi g-mean yang dihasilkan oleh algoritma SMOTEBoost-SVM dalam klasifikasi data seiring bertambahnya tingkat rasio *imbalance* pada data juga cenderung lebih kecil dibandingkan ketiga metode lain. Hal ini menunjukan bahwa SMOTEBoost menghasilkan nilai-nilai g-mean yang lebih stabil pada 20 replikasi yang dilakukan dibandingkan ketiga metode lain. Algoritma *boosting* lain yakni AdaBoost-SVM yang digunakan dalam penelitian ini menghasilkan performansi g-mean yang cukup baik dalam mengklasifikasikan data dengan rasio *balance* hingga rasio *imbalance* 5, namun performansinya menurun drastis ketika rasio *imbalance* pada data lebih dari 5.
2. Pada klasifikasi data *microarray* yakni data kanker colon dengan rasio *imbalance* 1,82 dan myeloma dengan rasio *imbalance* 3,8 dengan melakukan seleksi fitur terlebih dahulu, diperoleh hasil yang sejalan dengan studi simulasi, dimana SMOTEBoost unggul dalam nilai performansi g-mean dan *sensitivity*. Efek fitur seleksi juga diihat dalam analisis menggunakan SVM dimana diperoleh hasil bahwa menggunakan fitur-fitur informatif hasil dari penyeleksian fitur menggunakan FCBF, menghasilkan performansi yang lebih baik dibandingkan dengan menggunakan seluruh fitur pada klasifikasi.

5.2 Saran

Berdasarkan hasil analisis dan kesimpulan yang diperoleh terdapat beberapa hal yang dapat disarankan pada penelitian selanjutnya, yakni:

1. Pada proses klasifikasi menggunakan *boosting*, pada penelitian ini tidak dapat dilakukan pengecekan efek dari seleksi fitur terhadap hasil klasifikasi dikarenakan kompleksitas algoritma dan proses *running* yang mungkin sangat memakan waktu sehingga diharapkan pada penelitian selanjutnya dapat dilakukan analisis lebih jauh mengenai efek dari seleksi fitur terhadap klasifikasi menggunakan *boosting* baik AdaBoost-SVM dan SMOTEBoost-SVM.
2. Pada penelitian ini, penanganan data *imbalance* dilakukan dengan SMOTE dan *boosting* yang mana merupakan metode yang berbasis pada *preprocessing* data (*sampling*), sehingga diharapkan pada penelitian selanjutnya dapat dilakukan perbandingan dengan penanganan data *imbalance* menggunakan metode lain yang berbasis pada pendekatan algoritma, seperti fuzzy-SVM, atau menggunakan *boosting* tetapi dengan *base classifier Weight*-SVM (WSVM) dimana bobot-bobot pengamatan hasil *update* di tiap iterasi *boosting* langsung dimasukan ke dalam fungsi tujuan yang akan di optimasi.
3. Pada penelitian ini studi simulasi yang didesain hanya berfokus pada rasio kelas *imbalance*, sehingga selanjutnya diharapkan dapat mempertimbangkan efek dari jumlah pengamatan dam banyak-nya fitur yang digunakan dalam mellihat performansi dari algoritma *boosting*.

DAFTAR PUSTAKA

Abraham, R., Simha, J. B., dan Iyengar, S. S. (2009). Effective Discretization and Hybrid feature selection using Naïve Bayesian classifier for Medical datamining. *International Journal of Computational Intelligence Research*, *5*(2), 116-129.

Akbani, R., Kwek, S., dan Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. *Machine learning: ECML 2004*, 39-50.

Batista, G. E., Prati, R. C., dan Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, *6*(1), 20-29.

Batuwita, R., dan Palade, V. (2013). Class Imbalance Learning Methods for Support Vector Machine. Dalam d. M. Haibo He, *Imbalance Learning: Foundation, Algorithms, and Applications* (hal. 83-99). Berlin: John Wiley & Sons.

Bhattacharjee, A., Richards, W. G., Staunton, J., Li, C., Monti, S., Vasa, P., ... dan Loda, M. (2001). Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences*, *98*(24), 13790-13795.

Bolón-Canedo, V., Sánchez-Marono, N., Alonso-Betanzos, A., Benítez, J. M., dan Herrera, F. (2014). A review of microarray datasets and applied feature selection methods. *Information Sciences*, *282*, 111-135.

Bolón-Canedo, V., Sánchez-Maroño, N., dan Alonso-Betanzos, A. (2015). *Feature selection for high-dimensional data*. Springer.

Brown, M. P., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., ... dan Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, *97*(1), 262-267.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121-167.

Chawla, N. V., Bowyer, K. W., Hall, L. O., dan Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

Cui, Y., Ma, H., dan Saha, T. (2014). Improvement of power transformer insulation diagnosis using oil characteristics data preprocessed by smoteboost technique. *IEEE Transactions on Dielectrics and Electrical Insulation*, *21*(5), 2363-2373.

Demidova, L., Nikulchev, E., dan Sokolova, Y. (2016). The SVM Classifier Based on the Modified Particle Swarm Optimization. *arXiv preprint arXiv:1603.08296*.

Drummond, C., dan Holte, R. C. (2003). C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II* (Vol. 11). Washington DC: Citeseer.

Freund, Y., dan Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23-37). Springer, Berlin, Heidelberg.

Garcia, E., dan Lozano, F. (2007). Boosting Support Vector Machines. *MLDM Posters* , 153-167.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., ... dan Bloomfield, C. D. (1999). Molecular classification of cancer:class discovery and class prediction by gene expression monitoring. *science*, *286*(5439), 531-537.

Gunn, S. (1998). Support Vector Machines for Classification and Regression. Technical Report.

Guo, H., Ma, J., dan Li, Z. (2016). An Improved AdaBoost-SVM Model Based on Sample Weights and Sampling Equilibrum. *Advances in Computer Science Research* , 34-38.

Guyon, I., Weston, J., Barnhill, S., dan Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, *46*(1), 389-422.

Han, J., Kamber, M., dan Pei, J. (2012). *Data Mining : Concepts and Technique 3rd edition.* USA: Morgan Kaufman.

Härdle, W.K., Prastyo, D.D. dan Hafner, C.M., 2014. Support Vector Machines with Evolutionary Model Selection for Default Prediction. *The Oxford Handbook of Applied Nonparametric and Semiparametric Econometrics and Statistics*, pp.346-373.

Hsu, C. W., Chang, C. C., dan Lin, C. J. (2003). A practical guide to support vector classification. Taiwan: National Taiwan University

Huang, M. L., Hung, Y. H., Lee, W. M., Li, R. K., dan Jiang, B. R. (2014). SVM-RFE based feature selection and Taguchi parameters optimization for multiclass SVM classifier. *The Scientific World Journal*, 1-10.

Kang, P., dan Cho, S. (2006). EUS SVMs:Ensemble of Under Sampled SVMs for Imbalance Problems. Dalam d. I.king, *Lecture Notes in Computer Sciences* (Vol. 4232, hal. 837-846). Berlin: Springer-Verlag.

Khaulasari, H. (2016). *Combine Sampling-Least Square Support Vector Machine untuk Klasifikasi Multi Class Imbalanced Data.* Program Pascasarjana, Institut Teknologi Sepuluh Nopember, Surabaya.

Kim, H. C., Pang, S., Je, H. M., Kim, D., dan Bang, S. Y. (2003). Constructing support vector machine ensemble. *Pattern recognition*, *36*(12), 2757-2767.

Kim, M. J., Kang, D. K., dan Kim, H. B. (2015). Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction. *Expert Systems with Applications*, *42*(3), 1074-1082.

Lavanya, C., Nandihini, M., Niranjana, R., dan Gunavathi, C. (2014). Classification of Microarray Data Based On Feature Selection Method. *Int J Innovative Res Sci Eng Technol*, *3*(1), 1-9.

Li, X., Wang, L., dan Sung, E. (2008). AdaBoost with SVM-based component classifiers. *Engineering Applications of Artificial Intelligence*, *21*(5), 785-795.

Lin, W. J., & Chen, J. J. (2012). Class-imbalanced classifiers for high-dimensional data. *Briefings in bioinformatics*, *14*(1), 13-26.

Liu, Y., An, A., dan Huang, X. (2006). Boosting Prediction Accuracy on Imbalanced Datasets with SVM Ensembles. In *PAKDD* (Vol. 6, pp. 107-118).

Mayhua-López, E., Gómez-Verdejo, V., dan Figueiras-Vidal, A. R. (2015). A new boosting design of Support Vector Machine classifiers. *Information Fusion*, *25*, 63-71.

Pomeroy, S. L., Tamayo, P., Gaasenbeek, M., Sturla, L. M., Angelo, M., McLaughlin, M. E., ... dan Allen, J. C. (2002). Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, *415*(6870), 436-442.

Purnami, S. W., Andari, S., & Pertiwi, Y. D. (2015). High-Dimensional Data Classification Based on Smooth Support Vector Machines. *Procedia Computer Science*, *72*, 477-484.

Purnami, S. W., dan Trapsilasiwi, R. K. (2017). SMOTE-Least Square Support Vector Machine for Classification of Multiclass Imbalanced Data. In *Proceedings of the 9th International Conference on Machine Learning and Computing* (pp. 107-111). ACM.

Saberkari, H., Shamsi, M., Golabi, F., Amoshahy, M. J., & Sedaaghi, M. H. (2014). Performance Study of Cancer Selection/Classification Algorithms Based on Microarray Data. *Applied Medical Informatics*, *35*(4), 1-10.

Santosa, Budi. (2007). *Data Mining: Teknik Pemanfaatan Data untuk Keperluan Bisnis.* Yogyakarta: Graha Ilmu.

Sain, H., dan Purnami, S.W. (2015). Combine Sampling Support Vector Machine for Imbalanced Data Classification. *Procedia Computer Science*, *72,* 771-780.

Schapire, R.E., dan Freund, Yoav. (2012). *Boosing : Foundations and Algorithms.* London: MIT Press.

Schapire,R.E., dan Freund, Yoav. (1999). Short Introduction to *Boosting*. *Journal of Japanese Society for Artificial Intelegence , 14*, 771-780.

Scholkopf, B., dan Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Massachusetts: MIT Press.

Shipp, M. A., Ross, K. N., Tamayo, P., Weng, A. P., Kutok, J. L., Aguiar, R. C., ... dan Ray, T. S. (2002). Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature medicine*, *8*(1), 68-74.

Slonim, D. K. (2002). From patterns to pathways: gene expression data analysis comes of age. *Nature genetics*, *32*(Supp), 502.

Sun, J., dan Li, H. (2012). Financial distress prediction using support vector machines: Ensemble vs. individual. *Applied Soft Computing*, *12*(8), 2254-2265.

Tan, P, N., Steinbach, M., dan Kumar, V. (2006). *Introduction to Data Mining.* USA: Pearson.

Trapsilasiwi, R. K. (2013). *Klasifikasi Multiclass untuk Imbalanced Data Menggunakan SMOTE Least Square Support Vector Machine.* Program Pascasarjana, Institut Teknologi Sepuluh Nopember, Surabaya.

Trevino, V., Falciani, F., dan Barrera-Saldaña, H. A. (2007). DNA microarrays: a powerful genomic tool for biomedical and clinical research. *Molecular Medicine*, *13*(9-10), 527.

Vanitha, C. D. A., Devaraj, D., dan Venkatesulu, M. (2015). Gene expression data classification using support vector machine and mutual information-based gene selection. *Procedia Computer Science*, *47*, 13-21.

Vapnik, V., dan Cortes, C. (1995). Support Vector Networks. *Machine Learning* , 273-297.

Vapnik, V.N. (1995). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

Veropoulos, K., Campbell, C., dan Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI* (pp. 55-60).

Wang, S., Li, D., Song, X., Wei, Y., dan Li, H. (2011). A feature selection method based on improved fisher's discriminant ratio for text sentiment classification. *Expert Systems with Applications*, *38*(7), 8696-8702.

Wickramaratna, J., Holden, S., & Buxton, B. (2001). Performance degradation in boosting. *Multiple Classifier Systems*, 11-21.

Wu, G., dan Chang, E. Y. (2003). Adaptive feature-space conformal transformation for imbalanced-data learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)* (pp. 816-823).

Yang, X., Song, Q., dan Wang, Y. (2007). A weighted support vector machine for data classification. *International Journal of Pattern Recognition and Artificial Intelligence*, *21*(05), 961-976.

Yu, L., dan Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (pp. 856-863).

Yu, L., Wang, S., Lai, K.K., dan Zhou, L. (2008). *Bio-Inspired Credit Risk Analysis.* Berlin: Springer-Verlag.

LAMPIRAN

Lampiran 1. *Syntax* fungsi AdaBoost-SVM

```
adaboostori=function(X,y,X.test,y.test,iterasi,kernel,gamma=NULL,cost){
  #kernel yang digunakan hanya rbf dan linear
  if(!is.matrix(X)) X=as.matrix(X)
  if(!is.matrix(X)) X.test=as.matrix(X.test)
  n=nrow(X)
  final.test <- rep(0, length(y.test))
  bobot=rep(1/n,n)
  final.pred=list()
  error=c()
  a=c()
  Sign=function(x,kelas.positif,kelas.negatif){
    tanda=ifelse(x>=0,kelas.positif,kelas.negatif)
    return(tanda)
  }
  for (i in 1:iterasi){
    if(i == 1) { samp=sample(nrow(X), nrow(X), replace = FALSE) }
# sampling dengan pengembalian pada iterasi ke dua dan seterusnya
   else if(i != 1) { samp=sample(nrow(X), nrow(X), replace = TRUE, prob =
bobot) }
# membuat trainning set baru
    X.train=X[samp,]
    row.names(X.train)=NULL
    y.train=y[samp]
    if (length(y.train[y.train==-1])==0|length(y.train[y.train==1])==0) {
      cat("y train hanya berisi satu kelas","\n")
      a[i]=0
      final.pred[[i]]=matrix(0,ncol=1,nrow=length(y.test))
      break}
    row.names(y.train)=NULL
#training svm
    if (kernel=="linear"){
      weight.svm=svm(x=X.train,y=y.train,scale=F,kernel=kernel,cost=cost)
    }
    else {

weight.svm=svm(x=X.train,y=y.train,scale=F,kernel=kernel,gamma=gamma,cost
=cost)
    }
#prediksi seluruh data trainning kemudian menghitung error
    pred=predict(weight.svm,X)
    error[i]=sum(bobot*ifelse(pred!=y,1,0))/sum(bobot)
    if (error[i]<=0.000001) { print("iterasi berhenti")
      cat("iterasi=",i,"\n")
      cat("error=",error[i],"\n")
  final.pred,final.pred)
      a[i]=0
      final.pred[[i]]=matrix(0,ncol=1,nrow=length(y.test))
      break }
else if (error[i]>=0.49999) {
a[i]=0 ;bobot=rep(1/n,n)}
else {
#menghitung bobot classifier
a[i]=(1/2)*log((1-error[i])/error[i])
#mengupdate bobot pengamatan
bobot=(bobot*exp(-a[i]*ifelse(pred!=y,-1,1)))/sum(bobot*exp(-
a[i]*ifelse(pred!=y,-1,1)))}
# mengklasifikasikan data testing
final.pred[[i]]=attr(predict(weight.svm,X.test,decision.values=TRUE),"dec
ision.values")
    if(colnames(final.pred[[i]])=="-1/1"){
      final.pred[[i]]=-final.pred[[i]]}
else { final.pred[[i]]=final.pred[[i]]}
```

Lampiran 1. *Syntax* fungsi AdaBoost-SVM (lanjutan)

```
a=a/sum(a)
  if(is.nan(a[1])) {a=rep(0,length(a))}
  a=as.list(a)
# menggabungkan hasil klasifikasi (final classifier)
  dd=lapply(1:length(final.pred),function(i){final.pred[[i]]*a[[i]]})
  final.test=do.call("cbind",dd)
  final.test=rowSums(final.test)
  prediksi.kelas=Sign(final.test,1,-1)
 hasil=list(bobot.final=bobot,fit.y=final.test,prediksi.y=prediksi.kelas)
  return(hasil)
}
```

Lampiran 2. *Syntax* fungsi SMOTEBoost-SVM

```
smote.boost=function(X,y,X.test,y.test,iterasi,kernel,gamma=NULL,cost,k,o
ver=100){
  ## kernel yang digunakan hanya rbf dan linear
  #kelas minor == 1
  if(!is.matrix(X)) X=as.matrix(X)
  if(!is.matrix(X)) X.test=as.matrix(X.test)
  n=nrow(X)
  data=data.frame(X=X,y=y)
  data$bobot=rep(1/n,n)
  final.pred=list()
  error=c()
  a=c()
    Sign=function(x,kelas.positif,kelas.negatif){
    tanda=ifelse(x>=0,kelas.positif,kelas.negatif)
    return(tanda)
  }
  for (i in 1:iterasi){
# memodifikasi bobot dengan menggunakan smote
    datamayor=data[data$y==-1,]
  g = reformulate(paste(colnames(data)[which(colnames(data) != "y" &
colnames(data) != "bobot")], collapse = "+"), response = "y")
    dataminorbaru=SMOTE(g,data,k=k,perc.over = over,perc.under=0)
    train=rbind(datamayor,dataminorbaru)
    train$bobot = train$bobot / sum(train$bobot)
    samp=sample(nrow(train), nrow(train), replace = TRUE, prob =
train$bobot)
    train$bobot= NULL
    X.train=train[samp,colnames(train)!="y"]
    row.names(train)=NULL
    y.train=train[samp,colnames(train)=="y"]
    row.names(y.train)=NULL
    if (nlevels(y.train)==1) { cat ("y train hanya berisi satu kelas")
      a[i]=0
      final.pred[[i]]=matrix(0,ncol=1,nrow=length(y.test))
      break }
    if (kernel=="linear"){
      weight.svm=svm(x=X.train,y=y.train,scale=F,kernel=kernel,cost=cost)
    }else {

weight.svm=svm(x=X.train,y=y.train,scale=F,kernel=kernel,gamma=gamma,cost
=cost)}

weight.svm=wsvm(x=X,y=y,scale=F,kernel="radial",gamma=gamma,cost=cost,cas
e.weights = bobot)
    pred=predict(weight.svm,X)
    error[i]=sum(data$bobot*ifelse(pred!=y,1,0))/sum(data$bobot)
    if (error[i]<=0.000001) { print("iterasi berhenti")
      cat("iterasi=",i,"\n")
      cat("error=",error[i],"\n")
      a[i]=0
      final.pred[[i]]=matrix(0,ncol=1,nrow=length(y.test))
      break
    } else if (error[i]>=0.49999) {a[i]=0} else {a[i]=(1/2)*log((1-
error[i])/error[i])}
    if (error[i]>=0.49999){data$bobot=rep(1/n,n)} else {
    data$bobot=(data$bobot*exp(-a[i]*ifelse(pred!=y,-
1,1)))/sum(data$bobot*exp(-a[i]*ifelse(pred!=y,-1,1)))
    }
final.pred[[i]]=attr(predict(weight.svm,X.test,decision.values=TRUE),"dec
ision.values")
    if(colnames(final.pred[[i]])=="-1/1"){
      final.pred[[i]]=-final.pred[[i]]} else
{final.pred[[i]]=final.pred[[i]]}
  }
    a=a/sum(a)
    if(is.nan(a[1])) {a=rep(0,length(a))}
    a=as.list(a)
```

Lampiran 2. *Syntax* fungsi SMOTEBoost-SVM (lanjutan)

```
dd=lapply(1:length(final.pred),function(i){final.pred[[i]]*a[[i]]})
    final.test=do.call("cbind",dd)
    final.test=rowSums(final.test)
  prediksi.kelas=Sign(final.test,1,-1)


hasil=list(bobot.final=data$bobot,fit.y=final.test,prediksi.y=prediksi.ke
las,error=error,a=a)
  return(hasil)
}
```

Lampiran 3. *Syntax* Klasifikasi Data Riil Menggunakan SVM

```
# mengambil data kanker colon
library("datamicroarray", lib.loc="~/R/win-library/3.3")
data("alon")
table(alon$y)
data<-alon$x

##scalling data colon
for (i in 1:nrow(data)){
  for (j in 1:(ncol(data))){
    alon$x[i,j]=((data[i,j]-min(data[,j]))/(max(data[,j])-
min(data[,j]))*(1-0))+0
  }}
alon$y=ifelse(alon$y=="n",1,-1)
dataalon=as.data.frame(cbind(x=alon$x,y=alon$y))
dataalon$y=as.factor(dataalon$y)

##seleksi fitur data colon
library(Biocomb)
seleksi=select.fast.filter(dataalon,disc.method="MDL",
threshold=0.05,attrs.nominal=numeric())
datafix=dataalon[,c(seleksi$NumberFeature,ncol(dataalon))]

##mengambil data myeloma
data("tian")
table(tian$y)
data<-tian$x

##scalling data myeloma
for (i in 1:nrow(data)){
  for (j in 1:(ncol(data))){
    tian$x[i,j]=((data[i,j]-min(data[,j]))/(max(data[,j])-
min(data[,j]))*(1-0))+0
  }}
tian$y=ifelse(tian$y=="MRI-no-lytic-lesion sample",1,-1)
datatian=as.data.frame(cbind(x=tian$x,y=tian$y))
datatian$y=as.factor(datatian$y)

##seleksi fitur data myeloma
seleksi=select.fast.filter(datatian,disc.method="MDL",
threshold=0.05,attrs.nominal=numeric())
datafix=datatian[,c(seleksi$NumberFeature,ncol(datatian))]


# fungsi untuk membagi data menggunakan stratifikasi 5-fold cv
stratified.cv=function(data,nfolds){
  library(caret)
  folds <- createFolds(factor(data[,ncol(data)]), k = nfolds, list =
FALSE)
  data.train=vector("list",nfolds)
  data.test=vector("list",nfolds)
  for (i in 1:nfolds){
    fold<-which(folds==i,arr.ind = TRUE)
    data.train[[i]]<-data[-fold,]
    data.test[[i]]<-data[fold,]
  }
  data=list(data.train,data.test)
  #list [[1]][[1:5]] == data train dan list [[2]][[5:10]] == data.test
  return(data)
}

#membagi data ke training testing
set.seed(543290)
datacoba=stratified.cv(datafix,5)

#running klasifikasi dengan kernel linier
sink("svm_linier.txt")
konfusi=vector("list",5)
```

Lampiran 3. *Syntax* Klasifikasi Data Riil Menggunakan SVM (lanjutan)

```
out=vector("list",5)
gmean=c()
akurasi=c()
cost=c(0.1,1,10,100)
for(j in 1:4){
  cat("svm linier cost=",cost[j],"\n")
  for (i in 1:5){
    svm.e1071=svm(x=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,kernel="linear",cost=cost[j],scale=FALSE, type="C-classification")
    prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])
    pred.decision=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],decision.values=T)
    probs<-attr(pred.decision,"decision.values")

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
    print(out[[i]])
    gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
  }
  cat("rata-rata gmean=",mean(gmean),"\n")
  cat("rata-rata akurasi=",mean(akurasi),"\n")
  tab=do.call("cbind",out)
  sen=apply(tab,1,mean)
  print(sen)
  cat("\n")
}
sink()

#running klasifikasi dengan kernel radial
sink("svm_radial.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
akurasi=c()
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
for(j in 1:length(cost)){
  for (k in 1:length(gamma)){
    cat("model svm radial pada pasangan
cost",cost[j],"gamma=",gamma[k],"\n")
    for (i in 1:5){
      svm.e1071=svm(x=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,kernel="radial",cost=cost[j],gamma=gamma[k],scale=FALSE, type="C-
classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])
      pred.decision=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],decision.values=T)
      probs<-attr(pred.decision,"decision.values")

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table
      cat("matriks konfusi","\n")
      print(konfusi[[i]])
```

Lampiran 3. *Syntax* Klasifikasi Data Riil Menggunakan SVM (lanjutan)

```
out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      print(out[[i]])
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    cat("akurasi=",mean(akurasi),"\n")
    cat("gmean",mean(gmean),"\n")
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    print(sen)
    cat("\n")
  }
}
sink()
```

Lampiran 4. *Syntax* Klasifikasi Data Riil Menggunakan SMOTE-SVM

```
# klasifikasi menggunakan SMOTE-SVM radial pada data colon
Library(DMwR)
sink("svm_radial_smote.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
auc=c()
akurasi=c()
data.balance=vector("list",5)
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
for(j in 1:length(cost)){
  for (k in 1:length(gamma)){
    cat("model svm radial pada pasangan
cost",cost[j],"gamma=",gamma[k],"\n")
    for (i in 1:5){

datamayor=datacoba[[1]][[i]][datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])
]==-1,]
      set.seed(as.numeric(paste0(5,3,2,i)))
      dataminorbaru=SMOTE(y~.,datacoba[[1]][[i]],k=5,perc.over =
100,perc.under=0)
      data.balance[[i]]=rbind(datamayor,dataminorbaru)
      svm.e1071=svm(x=data.balance[[i]][,-
ncol(data.balance[[i]])],y=data.balance[[i]][,ncol(data.balance[[i]])],ke
rnel="radial",cost=cost[j],gamma=gamma[k], scale=FALSE, type="C-
classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table
      cat("matriks konfusi","\n")
      print(konfusi[[i]])

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      print(out[[i]])
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    cat("akurasi=",mean(akurasi),"\n")
    cat("gmean",mean(gmean),"\n")
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    print(sen)
    cat("\n")
  }}
sink()


# klasifikasi menggunakan SMOTE-SVM radial pada data myeloma

sink("svm_radial_smote.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
auc=c()
akurasi=c()
data.balance=vector("list",5)
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
for(j in 1:length(cost)){
  for (k in 1:length(gamma)){
    cat("model svm radial pada pasangan
cost",cost[j],"gamma=",gamma[k],"\n")
```

Lampiran 4. *Syntax* Klasifikasi Data Riil Menggunakan SMOTE-SVM (lanjutan)

```
    for (i in 1:5){

datamayor=datacoba[[1]][[i]][datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])
]==-1,]
      dataminorbaru=SMOTE(y~.,datacoba[[1]][[i]],k=5,perc.over =
300,perc.under=0)
      data.balance[[i]]=rbind(datamayor,dataminorbaru)
      svm.e1071=svm(x=data.balance[[i]][,-
ncol(data.balance[[i]])],y=data.balance[[i]][,ncol(data.balance[[i]])],ke
rnel="radial",cost=cost[j],gamma=gamma[k], scale=FALSE, type="C-
classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])))$table
      cat("matriks konfusi","\n")
      print(konfusi[[i]])

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])))$byClass[c(1,2)]
      print(out[[i]])
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    cat("akurasi=",mean(akurasi),"\n")
    cat("gmean",mean(gmean),"\n")
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    print(sen)
    cat("\n")
  }}
sink()

# klasifikasi menggunakan SMOTE-SVM linier pada data colon
sink("svm_radial_smote.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
auc=c()
akurasi=c()
data.balance=vector("list",5)
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
for(j in 1:length(cost)){
  for (k in 1:length(gamma)){
    cat("model svm radial pada pasangan
cost",cost[j],"gamma=",gamma[k],"\n")
    for (i in 1:5){

datamayor=datacoba[[1]][[i]][datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])
]==-1,]
      dataminorbaru=SMOTE(y~.,datacoba[[1]][[i]],k=5,perc.over =
300,perc.under=0)
      data.balance[[i]]=rbind(datamayor,dataminorbaru)
      svm.e1071=svm(x=data.balance[[i]][,-
ncol(data.balance[[i]])],y=data.balance[[i]][,ncol(data.balance[[i]])],ke
rnel="radial",cost=cost[j],gamma=gamma[k], scale=FALSE, type="C-
classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])
```

Lampiran 4. *Syntax* Klasifikasi Data Riil Menggunakan SMOTE-SVM (lanjutan)

```
akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table
      cat("matriks konfusi","\n")
      print(konfusi[[i]])

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      print(out[[i]])
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    cat("akurasi=",mean(akurasi),"\n")
    cat("gmean",mean(gmean),"\n")
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    print(sen)
    cat("\n")
  }}
sink()


# klasifikasi menggunakan SMOTE-SVM linier pada data myeloma
sink("svm_linier_smote.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
akurasi=c()
data.balance=vector("list",5)
cost=c(0.1,1,10,100)
for(j in 1:4){
  cat("SMOTE svm linier cost=",cost[j],"\n")
  for (i in 1:5){

datamayor=datacoba[[1]][[i]][datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])
]==-1,]
    dataminorbaru=SMOTE(y~.,datacoba[[1]][[i]],k=5,perc.over =
300,perc.under=0)
    data.balance[[i]]=rbind(datamayor,dataminorbaru)
    svm.e1071=svm(x=data.balance[[i]][,-
ncol(data.balance[[i]])],y=data.balance[[i]][,ncol(data.balance[[i]])],ke
rnel="linear",cost=cost[j], scale=FALSE, type="C-classification")
    prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
    print(out[[i]])
    print(konfusi[[i]])
    gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
  }
  cat("rata-rata gmean=",mean(gmean),"\n")
  cat("rata-rata akurasi=",mean(akurasi),"\n")
  tab=do.call("cbind",out)
  sen=apply(tab,1,mean)
  print(sen)
  cat("\n")
}
sink()
```

Lampiran 5. *Syntax* Klasifikasi Data Riil Menggunakan AdaBoost-SVM

```
# klasifikasi menggunakan adaboost radial
sink("boosting_radial_ori.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
akurasi=c()
gmean.iter=c()
akurasi.iter=c()
iterasi=seq(1,20,by=1)
sens.iter=vector("list",5)
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
matriks.akur=vector("list",length(cost))
matriks.gm=vector("list",length(cost))
matriks.akurasi=matrix(0,nrow=length(gamma),ncol=length(iterasi))
matriks.gmean=matrix(0,nrow=length(gamma),ncol=length(iterasi))
for(l in 1:length(cost)){
  for (k in 1:length(gamma)){
    cat("model svm radial pada pasangan
cost",cost[l],"gamma=",gamma[k],"\n")
    for (j in 1:length(iterasi)){
      cat(j,"iterasi digunakan","\n")
      for (i in 1:5){
        cat("validasi ke-",i,"\n")
        adaboost=adaboostori(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="radial",cost=cost[l],gamma=gamma[k],iterasi=iterasi[j])
        prediksi.boost=adaboost$prediksi.y
        prediksi.boost=as.factor(prediksi.boost)
        levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
        gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
      }
      print(gmean)
      gmean.iter[j]=mean(gmean)
      print(akurasi)
      akurasi.iter[j]=mean(akurasi)
      print(out)
      #print(konfusi)
      tab=do.call("cbind",out)
      sen=apply(tab,1,mean)
      print(sen)
      sens.iter[[j]]=sen
    }
    cat("nilai akurasi 20 iterasi","\n")
    print(akurasi.iter)
    matriks.akurasi[k,]=akurasi.iter
    cat("nilai gmean 20 iterasi","\n")
    print(gmean.iter)
    matriks.gmean[k,]=gmean.iter
    cat("nilai sensitivity dan specificity 20 iterasi maks","\n")
    print(sens.iter)
  }
  matriks.akur[[l]]=matriks.akurasi
  matriks.gm[[l]]=matriks.gmean
}
cat("hasil akurasi untuk setiap pasangan cost gamma maks 20
iterasi","\n")
```

Lampiran 5. *Syntax* Klasifikasi Data Riil Menggunakan AdaBoost-SVM (lanjutan)

```
print(matriks.akur)
cat("hasil gmean untuk setiap pasangan cost gamma maks 20 iterasi","\n")
print(matriks.gm)
sink()

# klasifikasi menggunakan adaboost linier

sink("boosting_linier_ori.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
akurasi=c()
gmean.iter=c()
akurasi.iter=c()
sens.iter=vector("list",5)
iterasi=seq(1,20,by=1)
matriks.akurasi=matrix(0,nrow=4,ncol=20)
matriks.gmean=matrix(0,nrow=4,ncol=20)
cost=c(0.1,1,10,100)
for(l in 1:length(cost)){
  cat("model svm linear pada cost",cost[l],"\n")
  for (j in 1:length(iterasi)){
    cat(j," iterasi","\n")
    for (i in 1:5){
      cat("validasi ke-",i,"\n")
      adaboost=adaboostori(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="linear",cost=cost[l],iterasi=iterasi[j])
      prediksi.boost=adaboost$prediksi.y
      prediksi.boost=as.factor(prediksi.boost)
      levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    print(gmean)
    gmean.iter[j]=mean(gmean)
    print(akurasi)
    akurasi.iter[j]=mean(akurasi)
    print(out)
    #print(konfusi)
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    print(sen)
    sens.iter[[j]]=sen
  }
  cat("nilai akurasi tiap iterasi","\n")
  print(akurasi.iter)
  matriks.akurasi[l,]=akurasi.iter
  cat("nilai gmean tiap iterasi","\n")
  print(gmean.iter)
  matriks.gmean[l,]=gmean.iter
  cat("nilai sensitivity dan specificity tiap iterasi","\n")
  print(sens.iter)
}
cat("hasil akurasi untuk setiap tiap cost maks 20 iterasi","\n")
print(matriks.akurasi)
```

Lampiran 5. *Syntax* Klasifikasi Data Riil Menggunakan AdaBoost-SVM (lanjutan)

```
cat("hasil gmean untuk setiap tiap cost maks 20 iterasi","\n")
print(matriks.gmean)
sink()
```

Lampiran 6. *Syntax* Klasifikasi Data Riil Menggunakan SMOTEBoost-SVM

```
# klasifikasi menggunakan SMOTEBoost radial
sink("smote_boosting_radial_ori.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
akurasi=c()
gmean.iter=c()
akurasi.iter=c()
iterasi=seq(1,20,by=1)
sens.iter=vector("list",5)
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
matriks.akur=vector("list",length(cost))
matriks.gm=vector("list",length(cost))
matriks.akurasi=matrix(0,nrow=length(gamma),ncol=length(iterasi))
matriks.gmean=matrix(0,nrow=length(gamma),ncol=length(iterasi))
for(l in 1:length(cost)){
  for (k in 1:length(gamma)){
    cat("model svm radial pada pasangan
cost",cost[l],"gamma=",gamma[k],"\n")
    for (j in 1:length(iterasi)){
      cat(j,"iterasi digunakan","\n")
      for (i in 1:5){
        cat("validasi ke-",i,"\n")
        smoteboost=smote.boost(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="radial",cost=cost[l],gamma=gamma[k],iterasi=iterasi[j],k=5,
over=100)
        prediksi.boost=smoteboost$prediksi.y
        prediksi.boost=as.factor(prediksi.boost)
        levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
        gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
      }
      print(gmean)
      gmean.iter[j]=mean(gmean)
      print(akurasi)
      akurasi.iter[j]=mean(akurasi)
      print(out)
      #print(konfusi)
      tab=do.call("cbind",out)
      sen=apply(tab,1,mean)
      print(sen)
      sens.iter[[j]]=sen
    }
    cat("nilai akurasi 20 iterasi","\n")
    print(akurasi.iter)
    matriks.akurasi[k,]=akurasi.iter
    cat("nilai gmean 20 iterasi","\n")
    print(gmean.iter)
    matriks.gmean[k,]=gmean.iter
    cat("nilai sensitivity dan specificity 20 iterasi maks","\n")
    print(sens.iter)
  }
  matriks.akur[[l]]=matriks.akurasi
  matriks.gm[[l]]=matriks.gmean
}
```

Lampiran 6. *Syntax* Klasifikasi Data Riil Menggunakan SMOTEBoost-SVM (lanjutan)

```
cat("hasil akurasi untuk setiap pasangan cost gamma maks 20
iterasi","\n")
print(matriks.akur)
cat("hasil gmean untuk setiap pasangan cost gamma maks 20 iterasi","\n")
print(matriks.gm)
sink()


# klasifikasi menggunakan SMOTEBoost linier
sink("smote_boosting_linier_ori.txt")
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
akurasi=c()
gmean.iter=c()
akurasi.iter=c()
sens.iter=vector("list",5)
iterasi=seq(1,20,by=1)
matriks.akurasi=matrix(0,nrow=4,ncol=20)
matriks.gmean=matrix(0,nrow=4,ncol=20)
cost=c(0.1,1,10,100)
for(l in 1:length(cost)){
  cat("model svm linear pada cost",cost[l],"\n")
  for (j in 1:length(iterasi)){
    cat(j," iterasi","\n")
    for (i in 1:5){
      cat("validasi ke-",i,"\n")
      smoteboost=smote.boost(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="linear",cost=cost[l],iterasi=iterasi[j],k=5,over=200)
      prediksi.boost=smoteboost$prediksi.y
      prediksi.boost=as.factor(prediksi.boost)
      levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    print(gmean)
    gmean.iter[j]=mean(gmean)
    print(akurasi)
    akurasi.iter[j]=mean(akurasi)
    print(out)
    #print(konfusi)
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    print(sen)
    sens.iter[[j]]=sen
  }
```

Lampiran 6. *Syntax* Klasifikasi Data Riil Menggunakan SMOTEBoost-SVM (lanjutan)

```
  cat("nilai akurasi tiap iterasi","\n")
  print(akurasi.iter)
  matriks.akurasi[l,]=akurasi.iter
  cat("nilai gmean tiap iterasi","\n")
  print(gmean.iter)
  matriks.gmean[l,]=gmean.iter
  cat("nilai sensitivity dan specificity tiap iterasi","\n")
  print(sens.iter)
}

cat("hasil akurasi untuk setiap tiap cost maks 20 iterasi","\n")
print(matriks.akurasi)
cat("hasil gmean untuk setiap tiap cost maks 20 iterasi","\n")
print(matriks.gmean)
sink()
```

Lampiran 7. *Syntax* Pada Studi Simulasi

```
# fungsi pembangkitan data simulasi
simulasi=function(n,rho,muminor,mumayor,rasio){
  library(MASS)
  dataminor=mvrnorm(round((1/(rasio+1))*n),mu=muminor,Sigma=rho)
  datamayor=mvrnorm(round((rasio/(rasio+1))*n),mu=mumayor,Sigma=rho)
  data=as.data.frame(rbind(cbind(datamayor,rep(-
1,nrow(datamayor))),cbind(dataminor,rep(1,nrow(dataminor)))))
  return(data)
}

# proses pembangkitan data simulasi rasio balance 20 replikasi
data("alon")
a=cor(alon$x)
set.seed(111)
indeks=sample(2000,100)
muminor=rep(0,2000)
muminor[indeks]=rep(1,100)
mumayor=rep(0,2000)
ulangan=20
data.simul=vector("list",ulangan)
datafix=vector("list",ulangan)
# data simulasi 1:1
for (u in 1:ulangan){
data.simul[[u]]=simulasi(n=100,rho=a,muminor=muminor,mumayor=mumayor,rasi
o=1)
  names(data.simul[[u]])=c(paste("x",1:2000),"y")
  data.simul[[u]]$y=as.factor(data.simul[[u]]$y)
  seleksi=select.fast.filter(data.simul[[u]],disc.method="MDL",
threshold=0.05,attrs.nominal=numeric())
datafix[[u]]=data.simul[[u]][,c(seleksi$NumberFeature,ncol(data.simul[[u]
]))]
}

# proses pembangkitan data simulasi rasio imbalance 2 20 replikasi

data("alon")
a=cor(alon$x)
set.seed(111)
indeks=sample(2000,100)
muminor=rep(0,2000)
muminor[indeks]=rep(1,100)
mumayor=rep(0,2000)
ulangan=20
data.simul=vector("list",ulangan)
datafix=vector("list",ulangan)
# data simulasi 1:1
for (u in 1:ulangan){

data.simul[[u]]=simulasi(n=100,rho=a,muminor=muminor,mumayor=mumayor,rasi
o=2)
  names(data.simul[[u]])=c(paste("x",1:2000),"y")
  data.simul[[u]]$y=as.factor(data.simul[[u]]$y)
  seleksi=select.fast.filter(data.simul[[u]],disc.method="MDL",
threshold=0.05,attrs.nominal=numeric())

datafix[[u]]=data.simul[[u]][,c(seleksi$NumberFeature,ncol(data.simul[[u]
]))]
}

# proses pembangkitan data simulasi rasio imbalance 5 20 replikasi
data("alon")
a=cor(alon$x)
set.seed(111)
indeks=sample(2000,100)
muminor=rep(0,2000)
muminor[indeks]=rep(1,100)
mumayor=rep(0,2000)
ulangan=20
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
data.simul=vector("list",ulangan)
datafix=vector("list",ulangan)
# data simulasi 1:1
for (u in 1:ulangan){

data.simul[[u]]=simulasi(n=100,rho=a,muminor=muminor,mumayor=mumayor,rasi
o=5)
  names(data.simul[[u]])=c(paste("x",1:2000),"y")
  data.simul[[u]]$y=as.factor(data.simul[[u]]$y)
  seleksi=select.fast.filter(data.simul[[u]],disc.method="MDL",
threshold=0.05,attrs.nominal=numeric())

datafix[[u]]=data.simul[[u]][,c(seleksi$NumberFeature,ncol(data.simul[[u]
]))]
}


# proses pembangkitan data simulasi rasio imbalance 10 20 replikasi
data("alon")

a=cor(alon$x)
set.seed(111)
indeks=sample(2000,100)
muminor=rep(0,2000)
muminor[indeks]=rep(1,100)
mumayor=rep(0,2000)
ulangan=20
data.simul=vector("list",ulangan)
datafix=vector("list",ulangan)
# data simulasi 1:1
for (u in 1:ulangan){

data.simul[[u]]=simulasi(n=100,rho=a,muminor=muminor,mumayor=mumayor,rasi
o=10)
  names(data.simul[[u]])=c(paste("x",1:2000),"y")
  data.simul[[u]]$y=as.factor(data.simul[[u]]$y)
  seleksi=select.fast.filter(data.simul[[u]],disc.method="MDL",
threshold=0.05,attrs.nominal=numeric())

datafix[[u]]=data.simul[[u]][,c(seleksi$NumberFeature,ncol(data.simul[[u]
]))]
}

# proses pembangkitan data simulasi rasio imbalance 15 20 replikasi
data("alon")
a=cor(alon$x)
set.seed(111)
indeks=sample(2000,100)
muminor=rep(0,2000)
muminor[indeks]=rep(1,100)
mumayor=rep(0,2000)
ulangan=20
data.simul=vector("list",ulangan)
datafix=vector("list",ulangan)
# data simulasi 1:1
for (u in 1:ulangan){

data.simul[[u]]=simulasi(n=100,rho=a,muminor=muminor,mumayor=mumayor,rasi
o=15)
  names(data.simul[[u]])=c(paste("x",1:2000),"y")
  data.simul[[u]]$y=as.factor(data.simul[[u]]$y)
  seleksi=select.fast.filter(data.simul[[u]],disc.method="MDL",
threshold=0.05,attrs.nominal=numeric())

datafix[[u]]=data.simul[[u]][,c(seleksi$NumberFeature,ncol(data.simul[[u]
]))]
}
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
# klasifikasi menggunakan SVM linier
sink("svm_linier_datasimul.txt")
ulangan=20
cost=c(0.1,1,10,100)
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
gmean.tiap.cost=c()
akurasi=c()
akurasi.tiap.cost=c()
sensitivity.tiap.cost=c()
specificity.tiap.cost=c()
g.mean.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
akurasi.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
sensitivity.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
specificity.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
for (u in 1:ulangan){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(j in 1:length(cost)){
    cat("svm linier cost=",cost[j],"\n")
    for (i in 1:length(datacoba[[1]])){
      svm.e1071=svm(x=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,kernel="linear",cost=cost[j],scale=FALSE, type="C-classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    gmean.tiap.cost[j]=mean(gmean)
    akurasi.tiap.cost[j]=mean(akurasi)
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    specificity.tiap.cost[j]=sen[1]
    sensitivity.tiap.cost[j]=sen[2]
    cat("\n")
  }
  g.mean.ulangan[u,]=gmean.tiap.cost
  akurasi.ulangan[u,]=akurasi.tiap.cost
  sensitivity.ulangan[u,]=sensitivity.tiap.cost
  specificity.ulangan[u,]=specificity.tiap.cost
}
row.names(g.mean.ulangan)=paste0("replikasi",1:ulangan)
colnames(g.mean.ulangan)=paste0(cost)
row.names(akurasi.ulangan)=paste0("replikasi",1:ulangan)
colnames(akurasi.ulangan)=paste0(cost)
row.names(sensitivity.ulangan)=paste0("replikasi",1:ulangan)
colnames(sensitivity.ulangan)=paste0(cost)
row.names(specificity.ulangan)=paste0("replikasi",1:ulangan)
colnames(specificity.ulangan)=paste0(cost)
cat("nilai gmean svm linier untuk setiap setiap replikasi","\n")
print(g.mean.ulangan)
write.csv2(t(g.mean.ulangan),paste0("gmean.csv"))
cat("nilai akurasi svm linier untuk setiap setiap replikasi","\n")
print(akurasi.ulangan)
write.csv2(t(akurasi.ulangan),paste0("akurasi.csv"))
cat("nilai specificity svm linier untuk setiap setiap replikasi","\n")
print(specificity.ulangan)
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
write.csv2(t(specificity.ulangan),paste0("spec.csv"))
cat("nilai sensitivity svm linier untuk setiap setiap replikasi","\n")
print(sensitivity.ulangan)
write.csv2(t(sensitivity.ulangan),paste0("sen.csv"))
sink()
# klasifikasi menggunakan SVM radial
sink("svm_radial_datasimul.txt")
ulangan=20
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
gmean.tiap.pasangan=c()
akurasi=c()
akurasi.tiap.pasangan=c()
sensitivity.tiap.pasangan=c()
specificity.tiap.pasangan=c()
kombinasi=expand.grid(cost,gamma)
g.mean.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
akurasi.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
sensitivity.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
specificity.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
for (u in 1:ulangan){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(j in 1:nrow(kombinasi)){
    cat("svm radial pasangan
cost=",kombinasi[j,1],"gamma=",kombinasi[j,2],"\n")
    for (i in 1:length(datacoba[[1]])){
      svm.e1071=svm(x=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,kernel="radial",cost=kombinasi[j,1],gamma=kombinasi[j,2],scale=FALSE,
type="C-classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    gmean.tiap.pasangan[j]=mean(gmean)
    akurasi.tiap.pasangan[j]=mean(akurasi)
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    specificity.tiap.pasangan[j]=sen[1]
    sensitivity.tiap.pasangan[j]=sen[2]
    cat("\n")
  }
  g.mean.ulangan[u,]=gmean.tiap.pasangan
  akurasi.ulangan[u,]=akurasi.tiap.pasangan
  sensitivity.ulangan[u,]=sensitivity.tiap.pasangan
  specificity.ulangan[u,]=specificity.tiap.pasangan
}
row.names(g.mean.ulangan)=paste0("replikasi",1:ulangan)
colnames(g.mean.ulangan)=paste0("pasangan",1:nrow(kombinasi))
row.names(akurasi.ulangan)=paste0("replikasi",1:ulangan)
colnames(akurasi.ulangan)=paste0("pasangan",1:nrow(kombinasi))
row.names(sensitivity.ulangan)=paste0("replikasi",1:ulangan)
colnames(sensitivity.ulangan)=paste0("pasangan",1:nrow(kombinasi))
row.names(specificity.ulangan)=paste0("replikasi",1:ulangan)
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
colnames(specificity.ulangan)=paste0("pasangan",1:nrow(kombinasi))
cat("nilai gmean svm radial untuk setiap setiap replikasi","\n")
print(g.mean.ulangan)
write.csv2(t(g.mean.ulangan),paste0("gmean.csv"))
cat("nilai akurasi svm radial untuk setiap setiap replikasi","\n")
print(akurasi.ulangan)
write.csv2(t(akurasi.ulangan),paste0("akurasi.csv"))
cat("nilai specificity svm radial untuk setiap setiap replikasi","\n")
print(specificity.ulangan)
write.csv2(t(specificity.ulangan),paste0("specificity.csv"))
cat("nilai sensitivity svm radial untuk setiap setiap replikasi","\n")
print(sensitivity.ulangan)
write.csv2(t(sensitivity.ulangan),paste0("sensitivity.csv"))
sink()

# klasifikasi menggunakan AdaBoost-SVM linier
sink("boosting_linier_ori_sim.txt")
konfusi=vector("list",5)
gmean=c()
akurasi=c()
sensitivity=c()
specificity=c()
gmean.iter=c()
akurasi.iter=c()
sens.iter=c()
spec.iter=c()
iterasi=seq(1,20,by=1)
akurasi.list=vector("list",20)
g.mean.list=vector("list",20)
sen.list=vector("list",20)
spec.list=vector("list",20)
matriks.akurasi=matrix(0,nrow=4,ncol=20)
matriks.gmean=matrix(0,nrow=4,ncol=20)
mat.sen=matrix(0,nrow=4,ncol=20)
mat.spec=matrix(0,nrow=4,ncol=20)
cost=c(0.1,1,10,100)
for (u in 1:20){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(l in 1:length(cost)){
    cat("model svm linear pada cost",cost[l],"\n")
    for (j in 1:length(iterasi)){
      cat(j," iterasi","\n")
      for (i in 1:5){
        cat("validasi ke-",i,"\n")
        adaboost=adaboostori(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="linear",cost=cost[l],iterasi=iterasi[j])
        prediksi.boost=adaboost$prediksi.y
        prediksi.boost=as.factor(prediksi.boost)
        levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

sensitivity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[1]

specificity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[2]
        gmean[i]=sqrt((sensitivity[i])*(specificity[i]))
      }
      gmean.iter[j]=mean(gmean)
      akurasi.iter[j]=mean(akurasi)
      sens.iter[j]=mean(sensitivity)
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
      spec.iter[j]=mean(specificity)
    }
    matriks.akurasi[l,]=akurasi.iter
    matriks.gmean[l,]=gmean.iter
    mat.sen[l,]=sens.iter
    mat.spec[l,]=spec.iter
  }
  row.names(matriks.akurasi)=paste0("cost",cost)
  colnames(matriks.akurasi)=paste0(iterasi,"iterasi maks")
  row.names(matriks.gmean)=paste0("cost",cost)
  colnames(matriks.gmean)=paste0(iterasi,"iterasi maks")
  row.names(mat.sen)=paste0("cost",cost)
  colnames(mat.sen)=paste0(iterasi,"iterasi maks")
  row.names(mat.spec)=paste0("cost",cost)
  colnames(mat.spec)=paste0(iterasi,"iterasi maks")
  write.csv2(matriks.akurasi,paste0("akurasi",u,".csv"))
  write.csv2(matriks.gmean,paste0("gmean",u,".csv"))
  write.csv2(mat.sen,paste0("specificity",u,".csv"))
  write.csv2(mat.spec,paste0("sensitivity",u,".csv"))
  akurasi.list[[u]]=matriks.akurasi
  g.mean.list[[u]]=matriks.gmean
  sen.list[[u]]=mat.sen
  spec.list[[u]]=mat.spec
}
print(akurasi.list)
print(g.mean.list)
print(sen.list)
print(spec.list)
cat("rata-rata 20 replikasi","\n")
aku=Reduce("+",akurasi.list)/20
gm=Reduce("+",g.mean.list)/20
se=Reduce("+",sen.list)/20
sp=Reduce("+",spec.list)/20
print(aku)
print(gm)
print(se)
print(sp)
sink()

# klasifikasi menggunakan AdaBoost-SVM radial
sink("boosting_radial_ori_sim.txt")
konfusi=vector("list",5)
gmean=c()
akurasi=c()
sensitivity=c()
specificity=c()
gmean.iter=c()
akurasi.iter=c()
sens.iter=c()
spec.iter=c()
iterasi=seq(1,20,by=1)
akurasi.list=vector("list",20)
g.mean.list=vector("list",20)
sen.list=vector("list",20)
spec.list=vector("list",20)
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
kombinasi=expand.grid(cost,gamma)
matriks.akurasi=matrix(0,nrow=nrow(kombinasi),ncol=20)
matriks.gmean=matrix(0,nrow=nrow(kombinasi),ncol=20)
mat.sen=matrix(0,nrow=nrow(kombinasi),ncol=20)
mat.spec=matrix(0,nrow=nrow(kombinasi),ncol=20)
for (u in 1:20){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(l in 1:nrow(kombinasi)){
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
    cat("model svm radial pada padangan
cost",kombinasi[l,1],"gamma",kombinasi[l,2],"\n")
    for (j in 1:length(iterasi)){
      cat(j," iterasi","\n")
      for (i in 1:5){
        cat("validasi ke-",i,"\n")
        adaboost=adaboostori(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="radial",cost=kombinasi[l,1],gamma=kombinasi[l,2],iterasi=it
erasi[j])
        prediksi.boost=adaboost$prediksi.y
        prediksi.boost=as.factor(prediksi.boost)
        levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

sensitivity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[1]

specificity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[2]
        gmean[i]=sqrt((sensitivity[i])*(specificity[i]))
      }
      gmean.iter[j]=mean(gmean)
      akurasi.iter[j]=mean(akurasi)
      sens.iter[j]=mean(sensitivity)
      spec.iter[j]=mean(specificity)
    }
    matriks.akurasi[l,]=akurasi.iter
    matriks.gmean[l,]=gmean.iter
    mat.sen[l,]=sens.iter
    mat.spec[l,]=spec.iter
  }
  row.names(matriks.akurasi)=paste0("pasangan cost
gamma",1:nrow(kombinasi))
  colnames(matriks.akurasi)=paste0(iterasi,"iterasi maks")
  row.names(matriks.gmean)=paste0("pasangan cost
gamma",1:nrow(kombinasi))
  colnames(matriks.gmean)=paste0(iterasi,"iterasi maks")
  row.names(mat.sen)=paste0("pasangan cost gamma",1:nrow(kombinasi))
  colnames(mat.sen)=paste0(iterasi,"iterasi maks")
  row.names(mat.spec)=paste0("pasangan cost gamma",1:nrow(kombinasi))
  colnames(mat.spec)=paste0(iterasi,"iterasi maks")
  write.csv2(matriks.akurasi,paste0("akurasi",u,".csv"))
  write.csv2(matriks.gmean,paste0("gmean",u,".csv"))
  write.csv2(mat.sen,paste0("specificity",u,".csv"))
  write.csv2(mat.spec,paste0("sensitivity",u,".csv"))
  akurasi.list[[u]]=matriks.akurasi
  g.mean.list[[u]]=matriks.gmean
  sen.list[[u]]=mat.sen
  spec.list[[u]]=mat.spec
}
print(akurasi.list)
print(g.mean.list)
print(sen.list)
print(spec.list)
cat("\n")
cat("rata-rata 20 replikasi","\n")
cat("\n")
aku=Reduce("+",akurasi.list)/20
gm=Reduce("+",g.mean.list)/20
se=Reduce("+",sen.list)/20
sp=Reduce("+",spec.list)/20
print(aku)
print(gm)
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
print(se)
print(sp)
sink()


# klasifikasi dengan smote-SVMlinier
# parameter smote berbeda beda untuk setiap tingkatan rasio imbalance
pada data. 100 untuk rasio 2, 400 untuk rasio 5, 900 untuk rasio 10, 1400
untuk rasio 15. K=5 kecuali pada data rasio 15 digunakan k=3.

sink("svm_linier_smote_datasimul.txt")
ulangan=20
cost=c(0.1,1,10,100)
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
gmean.tiap.cost=c()
akurasi=c()
akurasi.tiap.cost=c()
sensitivity.tiap.cost=c()
specificity.tiap.cost=c()
data.balance=vector("list",5)
g.mean.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
akurasi.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
sensitivity.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
specificity.ulangan=matrix(NA,nrow=ulangan,ncol=length(cost))
for (u in 1:ulangan){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(j in 1:length(cost)){
    cat("svm linier cost=",cost[j],"\n")
    for (i in 1:length(datacoba[[1]])){

datamayor=datacoba[[1]][[i]][datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])
]==-1,]
      dataminorbaru=SMOTE(y~.,datacoba[[1]][[i]],k=5,perc.over =
100,perc.under=0)
      data.balance[[i]]=rbind(datamayor,dataminorbaru)
      svm.e1071=svm(x=data.balance[[i]][,-
ncol(data.balance[[i]])],y=data.balance[[i]][,ncol(data.balance[[i]])],ke
rnel="linear",cost=cost[j], scale=FALSE, type="C-classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    gmean.tiap.cost[j]=mean(gmean)
    akurasi.tiap.cost[j]=mean(akurasi)
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    specificity.tiap.cost[j]=sen[1]
    sensitivity.tiap.cost[j]=sen[2]
    cat("\n")
  }
  g.mean.ulangan[u,]=gmean.tiap.cost
  akurasi.ulangan[u,]=akurasi.tiap.cost
  sensitivity.ulangan[u,]=sensitivity.tiap.cost
  specificity.ulangan[u,]=specificity.tiap.cost
}
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
row.names(g.mean.ulangan)=paste0("replikasi",1:ulangan)
colnames(g.mean.ulangan)=paste0(cost)
row.names(akurasi.ulangan)=paste0("replikasi",1:ulangan)
colnames(akurasi.ulangan)=paste0(cost)
row.names(sensitivity.ulangan)=paste0("replikasi",1:ulangan)
colnames(sensitivity.ulangan)=paste0(cost)
row.names(specificity.ulangan)=paste0("replikasi",1:ulangan)
colnames(specificity.ulangan)=paste0(cost)
cat("nilai gmean svm linier untuk setiap setiap replikasi","\n")
print(g.mean.ulangan)
write.csv2(t(g.mean.ulangan),paste0("gmean.csv"))
cat("nilai akurasi svm linier untuk setiap setiap replikasi","\n")
print(akurasi.ulangan)
write.csv2(t(akurasi.ulangan),paste0("akurasi.csv"))
cat("nilai specificity svm linier untuk setiap setiap replikasi","\n")
print(specificity.ulangan)
write.csv2(t(specificity.ulangan),paste0("spec.csv"))
cat("nilai sensitivity svm linier untuk setiap setiap replikasi","\n")
print(sensitivity.ulangan)
write.csv2(t(sensitivity.ulangan),paste0("sen.csv"))
sink()
# klasifikasi dengan smote-SVMradial
# parameter smote berbeda beda untuk setiap tingkatan rasio imbalance
pada data. 100 untuk rasio 2, 400 untuk rasio 5, 900 untuk rasio 10, 1400
untuk rasio 15. K=5 kecuali pada data rasio 15 digunakan k=3.

sink("svm_radial_smote_datasimul.txt")
ulangan=20
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
konfusi=vector("list",5)
out=vector("list",5)
gmean=c()
gmean.tiap.pasangan=c()
akurasi=c()
akurasi.tiap.pasangan=c()
sensitivity.tiap.pasangan=c()
specificity.tiap.pasangan=c()
data.balance=vector("list",5)
kombinasi=expand.grid(cost,gamma)
g.mean.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
akurasi.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
sensitivity.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
specificity.ulangan=matrix(NA,nrow=ulangan,ncol=nrow(kombinasi))
for (u in 1:ulangan){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(j in 1:nrow(kombinasi)){
    cat("svm radial pasangan
cost=",kombinasi[j,1],"gamma=",kombinasi[j,2],"\n")
    for (i in 1:length(datacoba[[1]])){

datamayor=datacoba[[1]][[i]][datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])
]==-1,]
      dataminorbaru=SMOTE(y~.,datacoba[[1]][[i]],k=3,perc.over =
100,perc.under=0)
      data.balance[[i]]=rbind(datamayor,dataminorbaru)
      svm.e1071=svm(x=data.balance[[i]][,-
ncol(data.balance[[i]])],y=data.balance[[i]][,ncol(data.balance[[i]])],ke
rnel="radial",cost=kombinasi[j,1],gamma=kombinasi[j,2],scale=FALSE,
type="C-classification")
      prediksi.e1071=predict(svm.e1071,datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])])

akurasi[i]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
konfusi[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,nco
l(datacoba[[2]][[i]])]))$table

out[[i]]=confusionMatrix(table(prediksi.e1071,datacoba[[2]][[i]][,ncol(da
tacoba[[2]][[i]])]))$byClass[c(1,2)]
      gmean[i]=sqrt((out[[i]][1])*(out[[i]][2]))
    }
    gmean.tiap.pasangan[j]=mean(gmean)
    akurasi.tiap.pasangan[j]=mean(akurasi)
    tab=do.call("cbind",out)
    sen=apply(tab,1,mean)
    specificity.tiap.pasangan[j]=sen[1]
    sensitivity.tiap.pasangan[j]=sen[2]
    cat("\n")
  }
  g.mean.ulangan[u,]=gmean.tiap.pasangan
  akurasi.ulangan[u,]=akurasi.tiap.pasangan
  sensitivity.ulangan[u,]=sensitivity.tiap.pasangan
  specificity.ulangan[u,]=specificity.tiap.pasangan
}
row.names(g.mean.ulangan)=paste0("replikasi",1:ulangan)
colnames(g.mean.ulangan)=paste0("pasangan",1:nrow(kombinasi))
row.names(akurasi.ulangan)=paste0("replikasi",1:ulangan)
colnames(akurasi.ulangan)=paste0("pasangan",1:nrow(kombinasi))
row.names(sensitivity.ulangan)=paste0("replikasi",1:ulangan)
colnames(sensitivity.ulangan)=paste0("pasangan",1:nrow(kombinasi))
row.names(specificity.ulangan)=paste0("replikasi",1:ulangan)
colnames(specificity.ulangan)=paste0("pasangan",1:nrow(kombinasi))
cat("nilai gmean svm radial untuk setiap setiap replikasi","\n")
print(g.mean.ulangan)
write.csv2(t(g.mean.ulangan),paste0("gmean.csv"))
cat("nilai akurasi svm radial untuk setiap setiap replikasi","\n")
print(akurasi.ulangan)
write.csv2(t(akurasi.ulangan),paste0("akurasi.csv"))
cat("nilai specificity svm radial untuk setiap setiap replikasi","\n")
print(specificity.ulangan)
write.csv2(t(specificity.ulangan),paste0("specificity.csv"))
cat("nilai sensitivity svm radial untuk setiap setiap replikasi","\n")
print(sensitivity.ulangan)
write.csv2(t(sensitivity.ulangan),paste0("sensitivity.csv"))
sink()


# klasifikasi dengan SMOTEBoost-SVM radial
# parameter smote berbeda beda untuk setiap tingkatan rasio imbalance
pada data. 100 untuk rasio 2, 400 untuk rasio 5, 900 untuk rasio 10, 1400
untuk rasio 15. K=5 kecuali pada data rasio 15 digunakan k=3.

sink("smoteboosting_radial_ori_sim.txt")
konfusi=vector("list",5)
gmean=c()
akurasi=c()
sensitivity=c()
specificity=c()
gmean.iter=c()
akurasi.iter=c()
sens.iter=c()
spec.iter=c()
iterasi=seq(1,20,by=1)
akurasi.list=vector("list",20)
g.mean.list=vector("list",20)
sen.list=vector("list",20)
spec.list=vector("list",20)
cost=c(0.1,1,10,100)
gamma=c(0.01,0.1,1,10)
kombinasi=expand.grid(cost,gamma)
matriks.akurasi=matrix(0,nrow=nrow(kombinasi),ncol=20)
matriks.gmean=matrix(0,nrow=nrow(kombinasi),ncol=20)
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
mat.sen=matrix(0,nrow=nrow(kombinasi),ncol=20)
mat.spec=matrix(0,nrow=nrow(kombinasi),ncol=20)
for (u in 1:20){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(l in 1:nrow(kombinasi)){
    cat("model svm radial pada padangan
cost",kombinasi[l,1],"gamma",kombinasi[l,2],"\n")
    for (j in 1:length(iterasi)){
      cat(j," iterasi","\n")
      for (i in 1:5){
        cat("validasi ke-",i,"\n")
        smoteboost=smote.boost(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="radial",cost=kombinasi[l,1],gamma=kombinasi[l,2],iterasi=it
erasi[j],k=5,over=100)
        prediksi.boost=smoteboost$prediksi.y
        prediksi.boost=as.factor(prediksi.boost)
        levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

sensitivity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[1]

specificity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[2]
        gmean[i]=sqrt((sensitivity[i])*(specificity[i]))
      }
      gmean.iter[j]=mean(gmean)
      akurasi.iter[j]=mean(akurasi)
      sens.iter[j]=mean(sensitivity)
      spec.iter[j]=mean(specificity)
    }
    matriks.akurasi[l,]=akurasi.iter
    matriks.gmean[l,]=gmean.iter
    mat.sen[l,]=sens.iter
    mat.spec[l,]=spec.iter
  }
  row.names(matriks.akurasi)=paste0("pasangan cost
gamma",1:nrow(kombinasi))
  colnames(matriks.akurasi)=paste0(iterasi,"iterasi maks")
  row.names(matriks.gmean)=paste0("pasangan cost
gamma",1:nrow(kombinasi))
  colnames(matriks.gmean)=paste0(iterasi,"iterasi maks")
  row.names(mat.sen)=paste0("pasangan cost gamma",1:nrow(kombinasi))
  colnames(mat.sen)=paste0(iterasi,"iterasi maks")
  row.names(mat.spec)=paste0("pasangan cost gamma",1:nrow(kombinasi))
  colnames(mat.spec)=paste0(iterasi,"iterasi maks")
  write.csv2(matriks.akurasi,paste0("akurasi",u,".csv"))
  write.csv2(matriks.gmean,paste0("gmean",u,".csv"))
  write.csv2(mat.sen,paste0("specificity",u,".csv"))
  write.csv2(mat.spec,paste0("sensitivity",u,".csv"))
  akurasi.list[[u]]=matriks.akurasi
  g.mean.list[[u]]=matriks.gmean
  sen.list[[u]]=mat.sen
  spec.list[[u]]=mat.spec
}
print(akurasi.list)
print(g.mean.list)
print(sen.list)
print(spec.list)
cat("\n")
cat("rata-rata 20 replikasi","\n")
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
cat("\n")
aku=Reduce("+",akurasi.list)/20
gm=Reduce("+",g.mean.list)/20
se=Reduce("+",sen.list)/20
sp=Reduce("+",spec.list)/20
print(aku)
print(gm)
print(se)
print(sp)
sink()


# klasifikasi dengan SMOTEBoost-SVM linier
# parameter smote berbeda beda untuk setiap tingkatan rasio imbalance
pada data. 100 untuk rasio 2, 400 untuk rasio 5, 900 untuk rasio 10, 1400
untuk rasio 15. K=5 kecuali pada data rasio 15 digunakan k=3.

sink("smoteboosting_linier_sim.txt")
konfusi=vector("list",5)
gmean=c()
akurasi=c()
sensitivity=c()
specificity=c()
gmean.iter=c()
akurasi.iter=c()
sens.iter=c()
spec.iter=c()
iterasi=seq(1,20,by=1)
akurasi.list=vector("list",20)
g.mean.list=vector("list",20)
sen.list=vector("list",20)
spec.list=vector("list",20)
cost=c(0.1,1,10,100)
matriks.akurasi=matrix(0,nrow=4,ncol=20)
matriks.gmean=matrix(0,nrow=4,ncol=20)
mat.sen=matrix(0,nrow=4,ncol=20)
mat.spec=matrix(0,nrow=4,ncol=20)
for (u in 1:20){
  cat("replikasi ke-",u,"\n")
  set.seed(21010)
  datacoba=stratified.cv(datafix[[u]],5)
  for(l in 1:4){
    cat("model svm linier pada cost",cost[l],"\n")
    for (j in 1:length(iterasi)){
      cat(j," iterasi","\n")
      for (i in 1:5){
        cat("validasi ke-",i,"\n")
        smoteboost=smote.boost(X=datacoba[[1]][[i]][,-
ncol(datacoba[[1]][[i]])],y=datacoba[[1]][[i]][,ncol(datacoba[[1]][[i]])]
,X.test=datacoba[[2]][[i]][,-
ncol(datacoba[[2]][[i]])],y.test=datacoba[[2]][[i]][,ncol(datacoba[[2]][[
i]])],kernel="linear",cost=cost[l],iterasi=iterasi[j],k=5,over=100)
        prediksi.boost=smoteboost$prediksi.y
        prediksi.boost=as.factor(prediksi.boost)
        levels(prediksi.boost)=c("-1","1")

akurasi[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,ncol(
datacoba[[2]][[i]])]))$overall[1]

sensitivity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[1]

specificity[i]=confusionMatrix(table(prediksi.boost,datacoba[[2]][[i]][,n
col(datacoba[[2]][[i]])]))$byClass[2]
        gmean[i]=sqrt((sensitivity[i])*(specificity[i]))
      }
      gmean.iter[j]=mean(gmean)
      akurasi.iter[j]=mean(akurasi)
```

Lampiran 7. *Syntax* Pada Studi Simulasi (lanjutan)

```
      sens.iter[j]=mean(sensitivity)
      spec.iter[j]=mean(specificity)
    }
    matriks.akurasi[l,]=akurasi.iter
    matriks.gmean[l,]=gmean.iter
    mat.sen[l,]=sens.iter
    mat.spec[l,]=spec.iter
  }
  row.names(matriks.akurasi)=paste0("cost",cost)
  colnames(matriks.akurasi)=paste0(iterasi,"iterasi maks")
  row.names(matriks.gmean)=paste0("cost",cost)
  colnames(matriks.gmean)=paste0(iterasi,"iterasi maks")
  row.names(mat.sen)=paste0("cost",cost)
  colnames(mat.sen)=paste0(iterasi,"iterasi maks")
  row.names(mat.spec)=paste0("cost",cost)
  colnames(mat.spec)=paste0(iterasi,"iterasi maks")
  write.csv2(matriks.akurasi,paste0("akurasi",u,".csv"))
  write.csv2(matriks.gmean,paste0("gmean",u,".csv"))
  write.csv2(mat.sen,paste0("specificity",u,".csv"))
  write.csv2(mat.spec,paste0("sensitivity",u,".csv"))
  akurasi.list[[u]]=matriks.akurasi
  g.mean.list[[u]]=matriks.gmean
  sen.list[[u]]=mat.sen
  spec.list[[u]]=mat.spec
}
print(akurasi.list)
print(g.mean.list)
print(sen.list)
print(spec.list)
cat("\n")
cat("rata-rata 20 replikasi","\n")
cat("\n")
aku=Reduce("+",akurasi.list)/20
gm=Reduce("+",g.mean.list)/20
se=Reduce("+",sen.list)/20
sp=Reduce("+",spec.list)/20
print(aku)
print(gm)
print(se)
print(sp)
sink()
```

Lampiran 8. Hasil Seleksi Fitur Menggunakan FCBF pada Data Riil

#data kanker colon

	Biomarker	Information.Gain	NumberFeature
1	X1671	0.3015691	1671
2	X249	0.2664472	249
3	X1772	0.2313463	1772
4	X625	0.2215022	625
5	X1042	0.2195625	1042
6	X1227	0.1699482	1227
7	X1153	0.1698471	1153
8	X467	0.1626530	467
9	X377	0.1584797	377
10	X1328	0.1273719	1328
11	X1473	0.1146799	1473
12	X279	0.1100181	279
13	X576	0.1100181	576
14	X682	0.1100181	682
15	X1560	0.1067025	1560

#data myeloma

	Biomarker	Information.Gain	NumberFeature
1	X35977_at	0.11573182	6034
2	X37451_at	0.10410546	7522
3	X34571_at	0.09827309	4614
4	AFFX.CreX.5_st	0.09425792	12576
5	X41098_at	0.09237511	11203
6	X39767_at	0.08564601	9860
7	X1178_at	0.08097466	196
8	X38413_at	0.07923651	8493
9	X37415_at	0.07917341	7486
10	X39377_at	0.07816757	9466
11	X32811_at	0.07707218	2837
12	X39794_at	0.07420942	9887
13	X34210_at	0.07145218	4250
14	X33731_at	0.06952555	3766
15	X34659_at	0.06893892	4703
16	X37360_at	0.06690387	7430
17	X40412_at	0.06618216	10511
18	X41758_at	0.06568935	11869
19	X38078_at	0.06479422	8155
20	X41660_at	0.06463249	11770
21	X1655_s_at	0.06250052	735
22	X31877_at	0.06250052	1893
23	X33729_at	0.06245617	3764
24	X38361_g_at	0.06113426	8441
25	X41530_at	0.06113426	11639
26	X34014_f_at	0.06097311	4052
27	X35482_at	0.06068404	5534
28	X40362_at	0.05967014	10460
29	X32616_at	0.05937948	2640
30	X39806_at	0.05851535	9899
31	X33738_r_at	0.05770292	3773

Lampiran 8. Hasil Seleksi Fitur Menggunakan FCBF pada Data Riil (lanjutan)

32	X36926_at	0.05759633	6993
33	X40787_at	0.05754402	10889
34	X32464_at	0.05690011	2486
35	X34714_at	0.05690011	4759
36	X39061_at	0.05690011	9147
37	X40801_at	0.05690011	10904
38	X1700_at	0.05689624	785
39	X38013_at	0.05667477	8090
40	X41220_at	0.05665568	11327
41	X39713_at	0.05661871	9806
42	X40448_at	0.05484050	10547
43	X39171_at	0.05468119	9258
44	X37529_at	0.05398292	7601
45	X39190_s_at	0.05358832	9277
46	X38491_at	0.05345725	8571
47	X31410_at	0.05329091	1422
48	X36107_at	0.05304848	6166
49	X33459_at	0.05268306	3491
50	X898_s_at	0.05237069	12460
51	X38350_f_at	0.05202812	8430
52	X31768_at	0.05183392	1783
53	X35003_at	0.05183392	5051
54	X38663_at	0.05161007	8745
55	X962_at	0.05161007	12524
56	X2047_s_at	0.05152472	1161
57	X41544_at	0.05077088	11653
58	X1664_at	0.05059454	745
59	X33379_at	0.05031098	3410

Lampiran 9. Performansi G-mean pada Klasifikasi data riil menggunakan AdaBoost-SVM

klasifikasi data alon

			iterasi									
Kernel	C	γ	1	2	3	...	...	18	19	20	maks	iter optimum
radial	0,1	0,01	0	0	0	...	...	0	0	0	0	-
	0,1	0,1	0	0	0	...	...	0	0	0	0	-
	0,1	1	0	0,31357	0,1	...	...	0,43086	0,45402	0,44586	0,63854	13
	0,1	10	0	0	0	...	...	0	0	0	0	-
	1	0,01	0	0	0	...	...	0	0	0	0	-
	1	0,1	0,67198	0,85611	0,85611	...	...	0,85611	0,86903	0,84156	0,89005	6
	1	1	0,86999	0,78984	0,79795	...	...	0,76011	0,80462	0,80103	0,86999	1
	1	10	-	-	-	...	...	-	-	-	-	-
	10	0,01	0,72438	0,85499	0,85611	...	...	0,85611	0,85611	0,85611	**0,89299**	16
	10	0,1	0,842234	0,80504	0,89126	...	...	0,80826	0,77460	0,82706	0,89126	3
	10	1	-	-	-	...	...	-	-	-	-	-
	10	10	-	-	-	...	...	-	-	-	-	-
	100	0,01	0,8662	0,82824	0,85418	...	...	0,77622	0,76182	0,81854	0,8662	1
	100	0,1	-	-	-	...	...	-	-	-	-	-
	100	1	-	-	-	...	...	-	-	-	-	-
	100	10	-	-	-	...	...	-	-	-	-	-
linier	0,1	-	0	0,82378	0,67146	...	...	0,88291	0,73296	0,81183	0,88291	18
	1	-	0,84026	0,79116	0,8662	...	...	0,80708	0,7919	0,86729	0,86729	20
	10	-	0,84223	0,85394	0,85035	...	...	0,76573	0,79317	0,84506	0,86903	5
	100	-	-	-	-	...	...	-	-	-	-	-

Lampiran 9. Performansi G-mean pada Klasifikasi data riil menggunakan AdaBoost-SVM (lanjutan)

klasifikasi data myeloma

			iterasi									
kernel	C	γ	1	2	3	...	...	18	19	20	maks	Iterasi optimum
radial	0,1	0,01	0	0	0	...	...	0	0	0	0	
	0,1	0,1	0	0,10498	0	...	...	0	0	0	0,10498	2
	0,1	1	0	0	0	...	...	0	0	0	0	
	0,1	10	0	0	0	...	...	0	0	0	0	
	1	0,01	0	0	0,07559	...	...	0,18057	0,36307	0,10691	0,39778	10
	1	0,1	0,43058	0,74254	0,64315	...	...	0,68643	0,70060	0,73979	0,78455	13
	1	1	-	-	-	...	...	-	-	-	-	
	1	10	-	-	-	...	...	-	-	-	-	
	10	0,01	0,55428	0,70652	0,54449	...	...	0,68972	0,72708	0,71113	**0,75619**	16
	10	0,1	0,73159	0,81024	0,72209	...	...	0,73549	0,72869	0,69719	0,81024	2
	10	1	-	-	-	...	...	-	-	-	-	
	10	10	-	-	-	...	...	-	-	-	-	
	100	0,01	0,78020	0,66458	0,71115	...	...	0,76008	0,72402	0,69540	0,78020	1
	100	0,1	0	0	0	...	...	0	0	0	0	
	100	1	-	-	-	...	...	-	-	-	-	
	100	10	-	-	-	...	...	-	-	-	-	
linier	0,1	-	0,07559	0,66683	0,69788	...	...	0,75976	0,78691	0,73666	0,78691	19
	1	-	0,76532	0,74532	0,71964	...	...	0,74006	0,73631	0,73247	0,76964	7
	10	-	-	-	-	...	...	-	-	-	-	
	100	-	-	-	-	...	...	-	-	-	-	

Lampiran 10. Performansi G-mean pada Klasifikasi data riil menggunakan SMOTEBoost-SVM

klasifikasi data alon

			Iterasi									
Kernel	C	γ	1	2	3	...	...	18	19	20	maks	iter optimum
radial	0,1	0,01	0	0	0	...	...	0	0	0	0,14142	11
	0,1	0,1	0	0	0	...	...	0	0	0	0	
	0,1	1	0,40956	0,90549	0,87439	...	...	0,89409	0,84506	0,85894	0,90549	2
	0,1	10	0	0	0	...	...	0	0,1	0	0,17321	9
	1	0,01	0	0	0	...	...	0	0,1	0,13229	0,14142	16
	1	0,1	0,85540	0,83833	0,88008	...	...	0,88008	0,85499	0,88008	0,88194	8
	1	1	0,82945	0,84102	0,81381	...	...	0,78594	0,76011	0,77479	0,84102	2
	1	10	0,66111	0,59213	0,56893	...	...	-	0,59771	-	0,72959	9
	10	0,01	0,80692	0,86229	0,82324	...	...	0,87617	0,84409	0,84111	**0,88008**	17
	10	0,1	0,88731	0,8662	0,82824	...	...	0,80705	0,81824	0,83102	0,88731	1
	10	1	0,81827	-	-	...	...	0,80103	-	0,81490	0,82945	14
	10	10	-	-	0,57337	...	...	0,63358	-	0,54414	0,66111	13
	100	0,01	0,88731	0,85111	0,87739	...	...	0,77123	0,78115	0,80595	0,88731	1
	100	0,1	0,76226	-	0,81714	...	...	0,82997	0,80705	0,80318	0,88409	14
	100	1	-	-	0,80318	...	...	-	0,76011	-	0,80318	3
	100	10	-	0,67233	0,72659	...	...	-	-	0,66111	0,72659	3
linier	0,1	-	0,67627	0,82362	0,83669	...	...	0,86806	0,84409	0,84409	0,89396	14
	1	-	0,80205	0,86229	0,84223	...	...	0,79205	0,82714	0,83712	0,89126	5
	10	-	0,86852	0,90514	0,84611	...	...	0,82214	0,84116	0,79317	0,90514	2
	100	-	0,81593	-	0,82832	...	...	0,76778	0,78198	0,85334	0,86512	5

Lampiran 10. Performansi G-mean pada Klasifikasi data riil menggunakan SMOTEBoost-SVM (lanjutan)

klasifikasi data myeloma

			Iterasi									
Kernel	C	γ	1	2	3	...	...	18	19	20	maks	iter optimum
radial	0,1	0,01	0,1361	0,0000	0,0000	...	...	0,0000	0,0000	0,0000	0,1361	1
	0,1	0,1	0,7726	0,5065	0,6715	...	...	0,6188	0,6678	0,6534	0,7726	1
	0,1	1	0,4315	0,3256	0,3896	...	...	0,2855	0,3303	0,3808	0,4502	4
	0,1	10	0,0000	0,0000	0,0000	...	...	0,0000	0,0000	0,0000	0,0000	
	1	0,01	0,8019	0,7395	0,7061	...	...	0,6257	0,7144	0,6724	0,8019	1
	1	0,1	0,7273	0,7037	0,8294	...	...	0,6800	0,7108	0,7259	0,8294	3
	1	1	0,5013	0,5833	0,5697	...	...	0,5872	0,6262	0,6191	0,6481	17
	1	10	0,0000	0,0000	0,0000	...	...	0,0000	0,0000	0,0000	0,0000	
	10	0,01	0,7716	0,8268	0,7663	...	...	0,7326	0,7928	0,7632	0,8268	2
	10	0,1	0,7564	0,7353	0,7531	...	...	0,6999	0,7013	0,7047	0,7564	1
	10	1	0,5373	0,5516	0,5564	...	...	0,6093	0,6137	0,6408	0,6727	8
	10	10	0,0000	0,0000	0,0000	...	...	0,0000	0,0000	0,0000	0,0000	
	100	0,01	0,7780	0,7987	0,7249	...	...	0,7442	0,7601	0,7567	0,7987	2
	100	0,1	0,7333	0,7728	0,7324	...	...	0,7033	0,7318	0,7013	0,7728	2
	100	1	0,5437	0,5500	0,5619	...	...	0,4649	-	0,5936	0,6593	11
	100	10	0,0000	0,0000	-	...	...	0,0000	0,0000	0,0000	0,1134	9
linier	0,1	-	0,7448	0,7355	0,7852	...	...	0,7250	0,7752	0,7904	0,8254	13
	1	-	0,8128	0,8320	0,6851	...	...	0,7540	0,7367	0,7429	0,8320	2
	10	-	0,7058	0,7879	0,7512	...	...	0,7525	0,7559	0,7409	0,7879	2
	100	-	0,6969	0,7192	0,7977	...	...	0,7165	0,7567	0,7601	0,7977	3

Lampiran 11. Performansi pada Klasifikasi data riil menggunakan SVM

klasifikasi data alon

model	C	gamma	akurasi	specificity	Sensitivity	g-mean
svm radial	0,1	0,01	0,6461538	1	0	0
	0,1	0,1	0,6461538	1	0	0
	0,1	1	0,6461538	1	0	0
	0,1	10	0,6461538	1	0	0
	1	0,01	0,6461538	1	0	0
	1	0,1	0,8089744	1	0,47	0,6719816
	1	1	0,8871795	0,925	0,83	**0,869989**
	1	10	0,6615385	0,95	0,14	0,2829842
	10	0,01	0,8397436	1	0,55	0,724376
	10	0,1	0,8538462	0,875	0,83	0,8422335
	10	1	0,8230769	0,875	0,75	0,7898391
	10	10	0,6615385	0,925	0,19	0,3695867
	100	0,01	0,8692308	0,875	0,87	0,8661996
	100	0,1	0,8397436	0,85	0,84	0,8333261
	100	1	0,8230769	0,875	0,75	0,7898391
	100	10	0,6615385	0,925	0,19	0,3695867
svm linier	0,1	-	0,6461538	1	0	0
	1	-	0,8705128	0,925	0,78	0,8402581
	10	-	0,8538462	0,875	0,83	0,8422335
	100	-	0,8230769	0,875	0,74	0,7902345

Lampiran 11. Performansi pada Klasifikasi data riil menggunakan SVM (lanjutan)

#klasifikasi data myeloma

Model	C	Gamma	akurasi	specificity	Sensitivity	gmean
svm radial	0,1	0,01	0,7919328	1	0	0
	0,1	0,1	0,7919328	1	0	0
	0,1	1	0,7919328	1	0	0
	0,1	10	0,7919328	1	0	0
	1	0,01	0,7919328	1	0	0
	1	0,1	0,8379832	0,98545	0,275	0,5068468
	1	1	0,7976471	0,98545	0,078571	0,1737236
	1	10	0,7919328	1	0	0
	10	0,01	0,8436975	0,98545	0,303571	0,530873
	10	0,1	0,8433613	0,934127	0,5	0,6562172
	10	1	0,8206723	0,956349	0,303571	0,4762329
	10	10	0,7919328	1	0	0
	100	0,01	0,854958	0,92672	0,582143	0,72338
	100	0,1	0,8436975	0,92672	0,528571	0,6901874
	100	1	0,8206723	0,956349	0,303571	0,4762329
	100	10	0,7919328	1	0	0
svm linier	0,1	-	0,7919328	0,992593	0,028571	0,07559289
	1	-	0,8668908	0,963757	0,503571	0,68444
	10	-	0,854958	0,912169	0,635714	**0,7519332**
	100	-	0,8378151	0,905027	0,578571	0,7121163

Lampiran 12. Performansi pada Klasifikasi data riil menggunakan SMOTE-SVM

#klasifikasi data alon

model	C	gamma	akurasi	specificity	Sensitivity	gmean
svm radial	0,1	0,01	0,353846	0	1	0
	0,1	0,1	0,353846	0	1	0
	0,1	1	0,8667	0,8	1	**0,8894**
	0,1	10	0,353846	0	1	0
	1	0,01	0,353846	0	1	0
	1	0,1	0,8526	0,85	0,87	0,853
	1	1	0,8705	0,9	0,83	0,8561
	1	10	0,6949	0,95	0,24	0,4179
	10	0,01	0,8692	0,85	0,92	0,8762
	10	0,1	0,8692	0,875	0,87	0,8662
	10	1	0,8231	0,875	0,75	0,7898
	10	10	0,6782	0,925	0,24	0,411
	100	0,01	0,8692	0,875	0,87	0,8662
	100	0,1	0,8551	0,85	0,88	0,8573
	100	1	0,8397	0,9	0,75	0,801
	100	10	0,6782	0,925	0,24	0,411
svm linier	0,1	-	0,8179	0,725	1	0,8447
	1	-	0,8538	0,875	0,83	0,8422
	10	-	0,8371	0,85	0,83	0,8302
	100	-	0,8231	0,875	0,74	0,7902

Lampiran 12. Performansi pada Klasifikasi data riil menggunakan SMOTE-SVM (lanjutan)

#klasifikasi data myeloma

Model	C	Gamma	akurasi	specificity	Sensitivity	gmean
svm radial	0,1	0,01	0,2081	0,0000	1,0000	0,0000
	0,1	0,1	0,7403	0,7019	0,8929	0,7877
	0,1	1	0,7919	0,9132	0,3357	0,4683
	0,1	10	0,2081	0,0000	1,0000	0,0000
	1	0,01	0,7928	0,7815	0,8143	0,7928
	1	0,1	0,8264	0,8466	0,7571	0,7947
	1	1	0,8150	0,9489	0,3036	0,5285
	1	10	0,7919	1,0000	0,0000	0,0000
	10	0,01	0,8323	0,8537	0,7571	0,7991
	10	0,1	0,8726	0,9415	0,6143	0,7505
	10	1	0,8091	0,9489	0,2750	0,4978
	10	10	0,7919	1,0000	0,0000	0,0000
	100	0,01	0,8437	0,9124	0,5857	0,7234
	100	0,1	0,8380	0,9196	0,5286	0,6878
	100	1	0,8150	0,9563	0,2750	0,4992
	100	10	0,7919	1,0000	0,0000	0,0000
svm linier	0,1	-	0,8326	0,8542	0,7571	0,7992
	1	-	0,8842	0,9270	0,7250	**0,8141**
	10	-	0,8435	0,9050	0,6071	0,7316
	100	-	0,8378	0,9050	0,5786	0,7121

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM

Pada bagian ini, akan diberikan ilustrasi mengenai proses *boosting* AdaBoost-SVM dengan 3 iterasi pada suatu data bangkitan. Data *training* ditampilkan sebagai berikut pada tabel berikut

pengamatan	x_1	x_2	y
1	0,149278	0,453972	-1
2	-1,30174	-1,84489	-1
3	-1,04878	0,303864	-1
4	0,181112	-0,85283	-1
5	-1,15326	-0,44438	-1
6	-0,70122	0,186564	-1
7	0,372285	1,488024	-1
8	0,589141	0,79688	1
9	1,079177	-0,17066	1
10	1,456053	1,142555	1

Diberikan pula data *testing* sebagai berikut:

pengamatan	x_1	x_2	y
11	0,222788	0,573344	-1
12	0,485774	-2,05008	1

Data *training* pada akan digunakan sebagai sebagai input dalam algoritma AdaBoost-SVM. Pada langkah awal *boosting* setiap data *training* diberi inisialisasi bobot. Bobot untuk setiap data *training* yakni $D_1(i)=1/m = 1/10$.

Iterasi :1

1. Membentuk data *training* baru berdasarkan bobot pengamatan.

   Pada awal proses iterasi, langkah pertama yang dilakukan yakni membentuk suatu data training baru yang akan digunakan dalam men-*training* SVM pada iterasi pertama. Data *training* baru dibentuk dengan melakukan proses sampling probabilitas tanpa pengembalian sebanyak m pengamatan, dimana bobot $D_1(i)$ akan digunakan sebagai probabilitas terpilihnya sampel. Sehingga diperoleh data *training* baru pada iterasi pertama *boosting* yakni

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)

Pengamatan	x_1	x_2	y
1	0,149278	0,453972	-1
2	-1,30174	-1,84489	-1
3	-1,04878	0,303864	-1
4	0,181112	-0,85283	-1
5	-1,15326	-0,44438	-1
6	-0,70122	0,186564	-1
7	0,372285	1,488024	-1
8	0,589141	0,79688	1
9	1,079177	-0,17066	1
10	1,456053	1,142555	1

Pada iterasi pertama data *training* baru akan sama dengan data *traning* awal, namun pada iterasi-iterasi selanjutnya data komposisi data akan berubah seiring berubahnya bobot pengamatan.

2. Melakukan training SVM pada data *training* baru

Selanjutnya data tersebut digunakan untuk melatih *weak learner* (SVM). pada ilustrasi ini akan digunakan fungsi kernel linier dengan parameter C=1. Fungsi klasifikasi SVM diperoleh dengan menyelesaikan persamaan

$$L_D(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j$$

dengan konstrain pada persamaan

$$0 \le a_i \le 1, \quad \sum_{i=1}^{n} a_i y_i = 0$$

Dengan bantuan solver libsvm pada software R diperoleh nilai-nilai solusi optimal untuk $\hat{\boldsymbol{\alpha}}$ yakni sebagai berikut.

$$\hat{\boldsymbol{\alpha}} = [0{,}4215 \quad 0 \quad 0 \quad 0{,}8779 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0{,}2993]^T$$

Pada vektor diatas dapat diketahui bahwa nilai-nilai yang tidak nol adalah indeks-indeks pengamatan yang merupakan *support vector*. Selanjutnya dengan mensubtitusi nilai $\hat{\boldsymbol{\alpha}}$ ke persamaan

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} a_i y_i \mathbf{x}_i$$

$$\hat{\mathbf{w}} = 0{,}4215(-1)\begin{bmatrix}0{,}149278\\0{,}453972\end{bmatrix} + \ldots + 0{,}2993(1)\begin{bmatrix}1{,}456053\\1{,}142555\end{bmatrix}$$

diperoleh

$$\hat{\mathbf{w}} = \begin{bmatrix}1{,}5099 & 0{,}0375\end{bmatrix}^T$$

Nilai $\hat{\boldsymbol{\alpha}}$ dan $\hat{\mathbf{w}}$ yang diperoleh kemudian disubtitusi ke persamaan

$$\hat{b} = \frac{1}{N_{sv}}\left(\sum_{i=1}^{N_{sv}} \frac{1}{y_i} - (\mathbf{w}^T \mathbf{x}_i)\right)$$

N_{sv} merupakan jumlah *support vector* yakni suatu pengamatan dengan nilai α_i berada pada selang $0 < \alpha_i \leq 1$. Diperoleh nilai bias $\hat{b} = $-1,2418. Lengkah selanjutnya yakni mengklasifikasikan setiap data *training* menggunakan fungsi pemisah yang diperoleh

$$\hat{f}(\mathbf{x}_i) = sign\,(\mathbf{x}_i^T\begin{bmatrix}1{,}5099\\0{,}0375\end{bmatrix} - 1{,}2418) \quad \begin{cases} +1\ jika & \mathbf{x}_i^T\begin{bmatrix}1{,}5099\\0{,}0375\end{bmatrix} - 1{,}2418 \geq 0 \\ -1\ jika & \mathbf{x}_i^T\begin{bmatrix}1{,}5099\\0{,}0375\end{bmatrix} - 1{,}2418 < 0 \end{cases}$$

Hasil klasifikasi data *training* yang diperoleh yaitu

$$\hat{\mathbf{y}} = \begin{bmatrix}-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1\end{bmatrix}^T$$

Dari hasil klasifikasi yang diperoleh, diketahui bahwa terdapat satu kesalahan klasifikasi yakni pada pengamatan ke-8. Ilustrasi hasil klasifikasi ditunjukan pada gambar dimana garis biru merupakan *hyperplane* (garis pemisah) $h_1(\mathbf{x}_i) = \mathbf{x}_i^T\begin{bmatrix}1{,}5099\\0{,}0375\end{bmatrix} - 1{,}2418$ yang diperoleh pada saat melakukan training weak classifier (SVM). Hubungan yang antara x_1 dan x_2 untuk membuat garis pemisah linier yakni

$$h(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = 0$$

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)

Sehingga jika disubtitusi ke persamaan tersebut

$$0 = [w_1 \quad w_2]^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b$$

$$0 = w_1 x_1 + w_2 x_2 + b$$

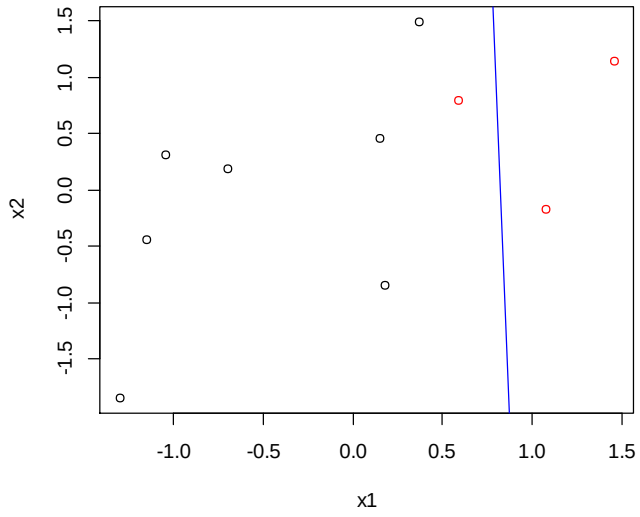
$$w_2 x_2 = -w_1 x_1 - b$$

$$x_2 = -\frac{w_1 x_1}{w_2} - \frac{b}{w_2}$$

$$x_2 = -\frac{1{,}5099 x_1}{0{,}0375} + \frac{1{,}2418}{0{,}0375}$$

$$x_2 = -40{,}264 x_1 + 33{,}11467$$

Persamaan tersebut kemudian digunakan untuk membuat garis pemisah seperti pada gambar berikut:



3. Melakukan perhitungan *error boosting*

   Perhitungan *error boosting* pada iterasi pertama dilakukan menggunakan persamaan

$$e_1 = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_1(i)$$

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)

Karena terdapat satu pengamatan yang nilai dari sign($h_1(\mathbf{x}_i)$)≠y_i (pengamatan tidak diklasifikasikan secara benar maka $\mathbf{e_1} = 0{,}1$.

4. Menghitung bobot *classifier*
   Bobot *classifier* dihitung menggunakan persamaan

$$a_1 = \frac{1}{2}\ln\left(\frac{1-e_1}{e_1}\right) = \frac{1}{2}\ln\left(\frac{0{,}9}{0{,}1}\right) = 1{,}098$$

5. Melakukan *update* bobot
   Bobot pengamatan selanjutnya di-*update* berdasarkan persamaan

$$D_2(i) = \frac{D_1(i)}{Z_1} \times \begin{cases} e^{-a_1} & jika\ sign(h_t(\mathbf{x}_i)) = y_i \\ e^{a_1} & jika\ sign(h_t(\mathbf{x}_i)) \neq y_i \end{cases}$$

   Dengan

$$Z_1 = \sum_{i=1}^{10} D_1(i) \times \begin{cases} e^{-a_1} & jika\ sign(h_1(\mathbf{x}_i)) = y_i \\ e^{a_1} & jika\ sign(h_1(\mathbf{x}_i)) \neq y_i \end{cases}$$

   Sehingga diperoleh vektor bobot yang nantinya akan digunakan pada iterasi ke dua yakni

$$\mathbf{D}_2 = [0.0556 \quad 0.0556 \quad 0.0556 \quad 0.0556 \quad 0.0556 \quad 0.0556 \quad 0.0556 \quad 0.5 \quad 0.0556 \quad 0.0556]^T$$

   bobot pemgamatan ke-8 diperbesar karena pengamatan tersebut sulit diklasifikasikan. Tujuan hal ini agar pada proses *training* selanjutnya *classifier* SVM akan lebih fokus pada pengamatan tersebut.

Iterasi 2:

Dengan alur yang sama seperti iterasi pertama, dibentuk data *training* baru yang akan digunakan untuk melatih SVM pada iterasi ke-dua. Pada iterasi kedua proses *sampling* dilakukan dengan cara *sampling* pengembalian sebanyak jumlah data *training* (m=10). Bobot yang telah di-*update* pada iterasi sebelumnya digunakan sebagai probabilitas terpilihnya sampel sehingga pengamatan ke-8 akan muncul lebih banyak dibandingkan pengamatan yang lain pada data *training* baru ini. Sepuluh Sampel yang terpilih yakni pengamatan ke-8, 8, 8, 1, 8, 4, 8, 6, 1, 5.

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)

Pengamatan	x_1	x_2	y
8	0,589141	0,79688	1
8	0,589141	0,79688	1
8	0,589141	0,79688	1
1	0,149278	0,453972	-1
8	0,589141	0,79688	1
4	0,181112	-0,85283	-1
8	0,589141	0,79688	1
6	-0,70122	0,186564	-1
1	0,149278	0,453972	-1
5	-1,15326	-0,44438	-1

Kemudian dilatih SVM dengan parameter C=1 menggunakan data *training* baru sehingga memperoleh

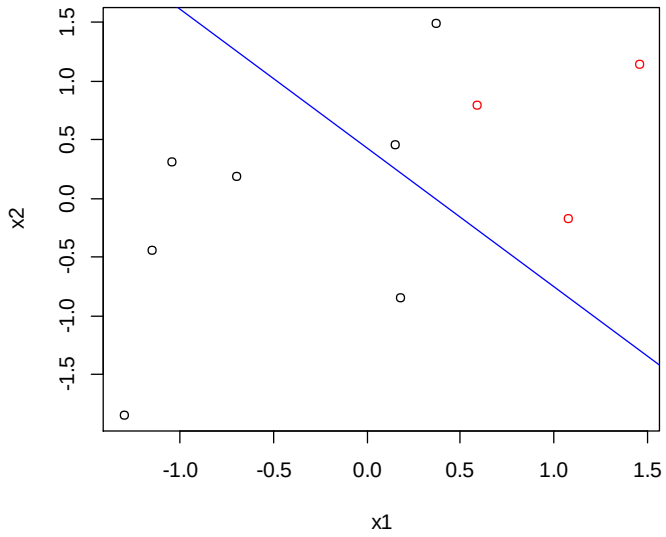
$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} 0 & 0 & 0{,}2792 & -1 & -1 & -0{,}1004 & -0{,}9647 & -0{,}1434 & -1 & 0 \end{bmatrix}^T$$

$$\hat{\mathbf{w}} = \begin{bmatrix} -1{,}1058 & -0{,}9391 \end{bmatrix}^T \text{ dan } \hat{f}(\mathbf{x}_i) = sign\,(\mathbf{x}_i^T \begin{bmatrix} -1{,}1058 \\ -0{,}9391 \end{bmatrix} + 0{,}3997) \ .$$

kemudian fungsi klasifikasi tersebut akan digunakan untuk mengklasifikasikan data *training* awal. Hasil klasifikasi yang diperoleh yakni sebagai berikut:

$\hat{\mathbf{y}} = \begin{bmatrix} 1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$ ilustrasi hasil klasifikasi ditunjukan pada gambar berikut:

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)



Garis biru merupakan *hyperplane*

$$h_2(\mathbf{x}_i) = \mathbf{x}_i^T \begin{bmatrix} -1{,}1058 \\ -0{,}9391 \end{bmatrix} + 0{,}3997 = 0 \rightarrow x_2 = -1{,}1775 x_1 + 0{,}4256$$

Berdasarkan gambar dapat diketahui bahwa pengamatan ke-8 dapat diklasifikasikan secara benar berkat pemberian bobot yang besar terhadap pengamatan tersebut. fungsi pemisah yang diperoleh (*hyperplane)* juga cenderung lebih *balance* (tidak condong ke kelas mayoritas) seperti pada fungsi pemisah pada iterasi pertama. Selanjutnya dihitung error training $\mathbf{e}_2 = 0{,}1111$ dan $a_2 = 1{,}0397$. Bobot pengamatan kemudian di-*update* dengan menggunakan nilai a_2 yang baru. Diperoleh bobot-bobot baru sebagai berikut

$$\mathbf{D}_3 = \begin{bmatrix} 0.25 & 0{,}0312 & 0{,}0312 & 0{,}0312 & 0{,}0312 & 0{,}0312 & 0{,}25 & 0{,}2813 & 0{,}0312 & 0{,}0312 \end{bmatrix}^T$$

Iterasi 3:

Dengan proses yang sama seperti iterasi-iterasi sebelumnya diperoleh 10 sampel yang terpilih yakni pengamatan ke-1, 8, 1, 7, 10, 7, 8, 8, 7, 8. Data *training* baru sebagai berikut

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)

pengamatan	x_1	x_2	y
1	0,149278	0,453972	-1
8	0,589141	0,79688	1
1	0,149278	0,453972	-1
7	0,372285	1,488024	-1
10	1,456053	1,142555	1
7	0,372285	1,488024	-1
8	0,589141	0,79688	1
8	0,589141	0,79688	1
7	0,372285	1,488024	-1
8	0,589141	0,79688	1

Proses training SVM pada data *trainning* menghasilkan :

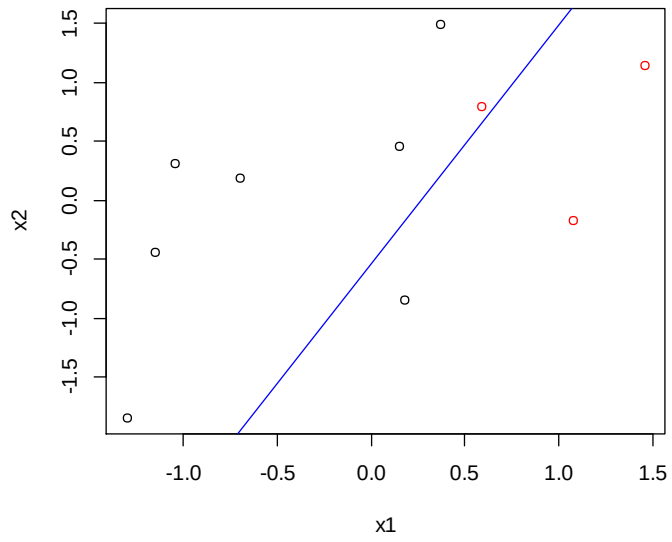
$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} 1 & 1 & 1 & 0{,}2596 & 0{,}2596 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

$$\hat{\mathbf{w}} = \begin{bmatrix} 1{,}5948 & -0{,}7862 \end{bmatrix}^T$$

$$\hat{f}(\mathbf{x}_i) = sign\,(\mathbf{x}_i^T \begin{bmatrix} 1{,}5948 \\ -0{,}7862 \end{bmatrix} + 0{,}4239)$$

$$\hat{\mathbf{y}} = \begin{bmatrix} -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \end{bmatrix}^T$$

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)



Dimana garis biru merupakan *hyperplane*

$$h_3(\mathbf{x}_i) = \mathbf{x}_i^T \begin{bmatrix} 1{,}5948 \\ -0{,}7862 \end{bmatrix} - 0{,}4239 = 0 \rightarrow x_2 = 2.0285x_1 - 0{,}4239$$

Selanjutnya dihitung *error training* $\mathbf{e}_3 = 0{,}3125$ dan $a_3 = 0{,}3942$. Setelah dilakukan proses iterasi sebnayak 3 iterasi diperoleh vektor bobot *classifier* $\mathbf{a} = [a_1 \quad a_2 \quad a_3]^T = [1{,}098 \quad 1{,}0397 \quad 0{,}3942]^T$.kemudian bobot *classifier* tersebut dinormalisasi sehingga diperoleh $\mathbf{a} = [0.4337 \quad 0{,}4106 \quad 0{,}1557]^T$. Final *classifier boosting* diperoleh dengan mengkombinasikan nilai bobot classifier dan fungsi pemisah yang diperoleh di tiap iterasi. *Final classifier* yang diperoleh yakni

$$H(\mathbf{x}_i) = sign\left(\begin{array}{l} a_1\left(\mathbf{x}_i^T \begin{bmatrix} 1{,}5099 \\ -0{,}0375 \end{bmatrix} - 1{,}2418 \right) + a_2\left(\mathbf{x}_i^T \begin{bmatrix} -1{,}1062 \\ -0{,}9383 \end{bmatrix} + 0{,}3996 \right) + \\ a_3\left(\mathbf{x}_i^T \begin{bmatrix} 1{,}5948 \\ -0{,}7862 \end{bmatrix} - 0{,}4239 \right) \end{array} \right)$$

final classifier tersebut akan digunakan untuk mengklasifikasikan data *testing*.

Lampiran 13. Ilustrasi Proses Klasifikasi Menggunakan AdaBoost-SVM (Lanjutan)

$$H(\mathbf{x}_1) = sign\left(\begin{array}{l} 0{,}4337\left(\begin{bmatrix}0{,}2228 & 0{,}5733\end{bmatrix}^T\begin{bmatrix}1{,}5099\\0{,}0375\end{bmatrix}-1{,}2418\right)+\\ 0{,}4106\left(\begin{bmatrix}0{,}2228 & 0{,}5733\end{bmatrix}^T\begin{bmatrix}-1{,}1058\\-0{,}9391\end{bmatrix}+0{,}3996\right)+\\ 0{,}1557\left(\begin{bmatrix}0{,}2228 & 0{,}5733\end{bmatrix}^T\begin{bmatrix}1{,}5948\\-0{,}7862\end{bmatrix}-0{,}4239\right)\end{array}\right) = sign(-0.6223) = -1$$

$$H(\mathbf{x}_2) = sign\left(\begin{array}{l} 0{,}4337\left(\begin{bmatrix}0{,}4858 & -2{,}0501\end{bmatrix}^T\begin{bmatrix}1{,}5099\\0{,}0375\end{bmatrix}-1{,}2418\right)+\\ 0{,}4106\left(\begin{bmatrix}0{,}4858 & -2{,}0501\end{bmatrix}^T\begin{bmatrix}-1{,}1058\\-0{,}9391\end{bmatrix}+0{,}3996\right)+\\ 0{,}1557\left(\begin{bmatrix}0{,}4858 & -2{,}0501\end{bmatrix}^T\begin{bmatrix}1{,}5948\\-0{,}7862\end{bmatrix}-0{,}4239\right)\end{array}\right) = sign(0.7858) = 1$$

Berdasarkan hasil tersebut, dapat diketahui bahwatidak terjadi kesalahan klasifikasi pada kedua data *testing*.

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM

Pada bagian ini akan diberikan ilustrasi mengenai proses iterasi SMOTEBoost-SVM dengan 2 iterasi pada salah satu contoh data bangkitan. Data *training* yang akan digunakan untuk membangun model *classifier* yakni sebeagi berikut

pengamatan	x_1	x_2	y
1	0,149278	0,453972	-1
2	-1,30174	-1,84489	-1
3	-1,04878	0,303864	-1
4	0,181112	-0,85283	-1
5	-1,15326	-0,44438	-1
6	-0,70122	0,186564	-1
7	0,372285	1,488024	-1
8	0,589141	0,79688	1
9	1,079177	-0,17066	1
10	1,456053	1,142555	1

Sementara data *testing* yang akan digunakan untuk men-*testing* model adalah sebagai berikut

pengamatan	x_1	x_2	y
11	0,222788	0,573344	-1
12	0,485774	-2,05008	1

Pada prinsipnya, prosedur iterasi dari algoritma ini adalah sama seperti algoritma Adaboost-SVM. pada awal iterasi diberikan inisialisai bobot untuk setiap pengamatan. Bobot untuk setiap pengamatan pada data *training* yakni $D_1(i)=1/m = 1/10$.

Iterasi 1:

1. Memodifikasi bobot pengamatan menggunakan algoritma SMOTE
   Pada langkah ini, akan dilakukan replikasi pengamatan-pengamatan pada kelas minoritas sebanyak 100 persen. Diketahui terdapat 3 pengamatan darri kelas minor (kelas positif). Setiap pengamatan tersebut akan mereplikasi sebanyak satu pengamatan sehingga nantinya akan diperoleh 3 pengamatan baru hasil replikasi. Langkah pertama pada proses replikasi yakni dicari k-tetangga terdekat dari masing masing pengamatan kelas minoritas. Karrena setiap pengamatan akan hanya memiliki 2 tetangga maka, hanya akan dicari 1 tetangga terdekat dari setiap pengamatan. Pada proses repliaksi ini, bobot pengamatan dimasukan sebagai variabel. Tetangga terdekat diperoleh dengan menggunakan jarak *euclidean* minimum.

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\left[\mathbf{x}_i - \mathbf{x}_j\right]^T \left[\mathbf{x}_i - \mathbf{x}_j\right]}$$

$$d(\mathbf{x}_8, \mathbf{x}_9) = \sqrt{\left[\mathbf{x}_8 - \mathbf{x}_9\right]^T \left[\mathbf{x}_8 - \mathbf{x}_9\right]} = 1{,}0846$$

$$d(\mathbf{x}_8, \mathbf{x}_{10}) = \sqrt{\left[\mathbf{x}_8 - \mathbf{x}_9\right]^T \left[\mathbf{x}_8 - \mathbf{x}_9\right]} = 0{,}9333$$

$$d(\mathbf{x}_9, \mathbf{x}_{10}) = \sqrt{\left[\mathbf{x}_8 - \mathbf{x}_9\right]^T \left[\mathbf{x}_8 - \mathbf{x}_9\right]} = 0{,}9333$$

Sehingga diperoleh tetangga terdekat dari pengamatan ke-8 yakni pengamatan ke-10, tetangga dari pengamatan ke-9 yakni pengamatan ke-8 dan tetangga terdekat dari pengamatan ke-10 yakni pengamatan ke-8. Maka data sintetis yang direplikasi berdasarkan pengamatan ke-8 diperoleh menggunakan persamaan (2.37)

$$\mathbf{x}_{syn1} = \mathbf{x}_8 + (\mathbf{x}_{10} - \mathbf{x}_8)\gamma = \begin{bmatrix} 0{,}589141 \\ 0{,}79688 \\ 0{,}1 \end{bmatrix} + \left(\begin{bmatrix} 1{,}456053 \\ 1{,}142555 \\ 0{,}1 \end{bmatrix} - \begin{bmatrix} 0{,}589141 \\ 0{,}79688 \\ 0{,}1 \end{bmatrix} \right).0{,}3261 = \begin{bmatrix} 0{,}8718407 \\ 0{,}9096046 \\ 0{,}1 \end{bmatrix}$$

dengan cara yang sama diperoleh data sintetis kedua dan ketiga yakni

$$\mathbf{x}_{syn2} = \mathbf{x}_9 + (\mathbf{x}_8 - \mathbf{x}_9)\gamma = \begin{bmatrix} 1{,}079177 \\ -0{,}17066 \\ 0{,}1 \end{bmatrix} + \left(\begin{bmatrix} 0{,}589141 \\ 0{,}79688 \\ 0{,}1 \end{bmatrix} - \begin{bmatrix} 1{,}079177 \\ -0{,}17066 \\ 0{,}1 \end{bmatrix} \right).0{,}8924 = \begin{bmatrix} 0{,}6418916 \\ 0{,}6927267 \\ 0{,}1 \end{bmatrix}$$

$$\mathbf{x}_{syn3} = \mathbf{x}_{10} + (\mathbf{x}_8 - \mathbf{x}_{10})\gamma = \begin{bmatrix} 1{,}456053 \\ 1{,}142555 \\ 0{,}1 \end{bmatrix} + \left(\begin{bmatrix} 0{,}589141 \\ 0{,}79688 \\ 0{,}1 \end{bmatrix} - \begin{bmatrix} 1{,}456053 \\ 1{,}142555 \\ 0{,}1 \end{bmatrix} \right).0{,}7013 = \begin{bmatrix} 0{,}8480512 \\ 0{,}9001187 \\ 0{,}1 \end{bmatrix}$$

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)

Sehingga diperoleh bobot dan pengamatan baru sebagai berikut

pengamatan	x1	x2	y	bobot
1	0,149278	0,453972	-1	0,1
2	-1,30174	-1,84489	-1	0,1
3	-1,04878	0,303864	-1	0,1
4	0,181112	-0,85283	-1	0,1
5	-1,15326	-0,44438	-1	0,1
6	-0,70122	0,186564	-1	0,1
7	0,372285	1,488024	-1	0,1
8	0,589141	0,79688	1	0,1
9	1,079177	-0,17066	1	0,1
10	1,456053	1,142555	1	0,1
11	0,871841	0,909605	1	0,1
12	0,641892	0,692727	1	0,1
13	0,848051	0,900119	1	0,1

Pada tabel diatas, pengamatan ke-11, 12, 13 merupakan data sintetis. Bobot-bobot tersebut kemudian dinormalisasi menjadi masing-masing pengamatan memiliki bobot $D_1(i)= 0,07692$

Langkah selanjutnya yaitu melakukan *sampling* dengan pengembalian sebanyak 13 sampel dengan menggunakan bobot baru tersebut sebagai probabilitas terpilihnya sampel. Sampel yang terpilih yakni pengamatan ke-4, 7, 7, 9, 12, 13, 6, 12, 12, 2, 13, 12,4.

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)

Sehingga diperoleh data *training* baru sebagai berikut

pengamatan	x1	x2	Y	bobot
4	0,181112	-0,85283	-1	0,076923
7	0,372285	1,488024	-1	0,076923
7	0,372285	1,488024	-1	0,076923
9	1,079177	-0,17066	1	0,076923
12	0,641892	0,692727	1	0,076923
13	0,848051	0,900119	1	0,076923
6	-0,70122	0,186564	-1	0,076923
12	0,641892	0,692727	1	0,076923
12	0,641892	0,692727	1	0,076923
2	-1,30174	-1,84489	-1	0,076923
13	0,848051	0,900119	1	0,076923
12	0,641892	0,692727	1	0,076923
4	0,181112	-0,85283	-1	0,076923

Pada data tersebut diperoleh pengamatan yang seimbang antara kelas positif dan negatif. Data *training* ini akan digunakan untuk melatih SVM. Proses *training* SVM

2. Melakukan *training* SVM pada data *training* baru
   pada ilustrasi ini akan digunakan fungsi kernel linier dengan parameter C=1. Fungsi klasifikasi SVM diperoleh dengan menyelesaikan persamaan

$$L_D(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j$$

dengan konstrain pada persamaan

$$0 \le a_i \le 1, \quad \sum_{i=1}^{n} a_i y_i = 0$$

Dengan bantuan *solver* libsvm pada software R diperoleh nilai-nilai solusi optimal untuk $\hat{\boldsymbol{\alpha}}$ yakni sebagai berikut.

$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} 1 & 1 & 1 & 0{,}5083 & 0{,}3396 & 0 & 0 & 1 & 1 & 0 & 0 & 0{,}992 & 0{,}8399 \end{bmatrix}^T$$

Selanjutnya dengan mensubtitusi nilai $\hat{\boldsymbol{\alpha}}$ ke persamaan $\hat{\mathbf{w}} = \sum_{i=1}^{n} a_i y_i \mathbf{x}_i$

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)

Sehingga diperoleh

$$\hat{\mathbf{w}} = \begin{bmatrix} 1{,}609283 & 0{,}8142541 \end{bmatrix}^T$$

Nilai $\hat{\boldsymbol{\alpha}}$ dan $\hat{\mathbf{w}}$ yang diperoleh kemudian disubtitusi ke persamaan

$$\hat{b} = \frac{1}{N_{sv}} \left(\sum_{i=1}^{N_{sv}} \frac{1}{y_i} - (\mathbf{w}^T \mathbf{x}_i) \right)$$

sehingga diperoleh nilai bias $\hat{b} = 0{,}5972$. Lengkah selanjutnya yakni mengklasifikasikan setiap data *training* awal menggunakan fungsi pemisah yang diperoleh

$$\hat{f}(\mathbf{x}_i) = sign\,(\mathbf{x}_i^T \begin{bmatrix} 1{,}609283 \\ 0{,}814254 \end{bmatrix} - 0{,}5972) \begin{cases} +1 \; jika & \mathbf{x}_i^T \begin{bmatrix} 1{,}609283 \\ 0{,}814254 \end{bmatrix} - 0{,}5972 \geq 0 \\ -1 \; jika & \mathbf{x}_i^T \begin{bmatrix} 1{,}609283 \\ 0{,}814254 \end{bmatrix} - 0{,}5972 < 0 \end{cases}$$

.

Hasil klasifikasi data *training* awal yang diperoleh yaitu

$$\hat{\mathbf{y}} = \begin{bmatrix} 1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

Dari hasil klasifikasi yang diperoleh, diketahui bahwa pengamatan kelas positif dapat diklasifikasikan secara benar. Ilustrasi hasil klasifikasi ditunjukan pada gambar dimana garis biru merupakan *hyperplane* (fungsi pemisah) $h_1(\mathbf{x}_i) = \mathbf{x}_i^T \begin{bmatrix} 1{,}609283 \\ 0{,}814254 \end{bmatrix} - 0{,}5972$ yang diperoleh pada saat melakukan *training weak classifier* (SVM).

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)

Hubungan yang antara variabel x_1 dan x_2 untuk membuat garis pemisah linier yakni

$$h(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b = 0$$

Sehingga jika disubtitusi ke persamaan tersebut

$$0 = [w_1 \quad w_2]^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b$$

$$0 = w_1 x_1 + w_2 x_2 + b$$

$$w_2 x_2 = -w_1 x_1 - b$$

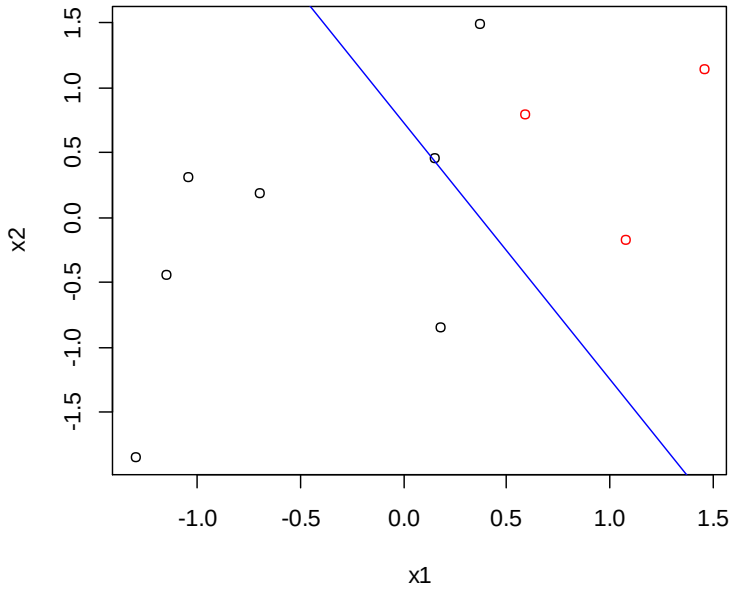
$$x_2 = -\frac{w_1 x_1}{w_2} - \frac{b}{w_2}$$

$$x_2 = -\frac{1{,}609283 x_1}{0{,}814254} + \frac{0.5972}{0{,}814254}$$

$$x_2 = -1{,}9764 x_1 + 0{,}7343$$

Persamaan tersebut kemudian digunakan untuk membuat garis pemisah seperti pada gambar berikut:

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)



3. Menghitung *error training*
   Perhitungan *error training* pada iterasi pertama *boosting* dilakukan menggunakan persamaan

$$e_1 = \sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_1(i)$$

   Karena terdapat dua pengamatan yang sign($h_1(\mathbf{x}_i)$)≠y_i (pengamatan tidak diklasifikasikan secara benar yakni pengamatan ke-1 dan 7. maka

$$\mathbf{e}_1 = 0{,}1 + 0{,}1 = 0{,}2$$

4. Menghitung bobot *classifier*
   Bobot *classifier* diperoleh dengan menggunakan persamaan

$$a_1 = \frac{1}{2}\ln\left(\frac{1-e_1}{e_1}\right) = \frac{1}{2}\ln\left(\frac{0{,}8}{0{,}2}\right) = 0{,}6931$$

Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)

6. Melakukan *update* bobot
   Bobot pengamatan selanjutnya di-*update* berdasarkan persamaan

$$D_2(i) = \frac{D_1(i)}{Z_1} \times \begin{cases} e^{-a_1} & jika\ sign(h_t(\mathbf{x}_i)) = y_i \\ e^{a_1} & jika\ sign(h_t(\mathbf{x}_i)) \neq y_i \end{cases}$$

   Dengan

$$Z_1 = \sum_{i=1}^{10} D_1(i) \times \begin{cases} e^{-a_1} & jika\ sign(h_1(\mathbf{x}_i)) = y_i \\ e^{a_1} & jika\ sign(h_1(\mathbf{x}_i)) \neq y_i \end{cases}$$

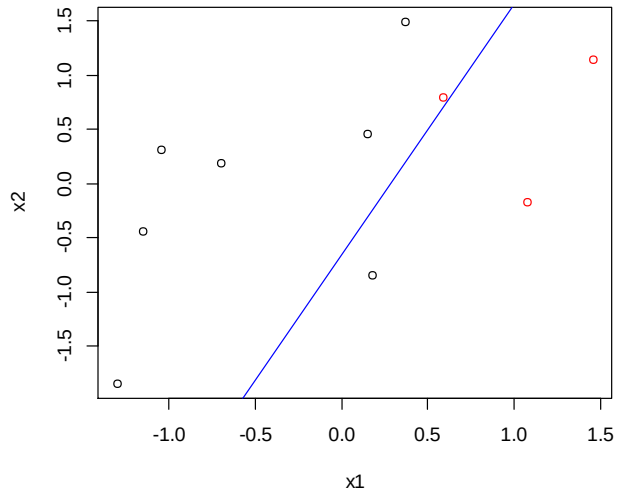
   Sehingga diperoleh vektor bobot baru seperti berikut:

$$\mathbf{D}_2 = [0{,}25 \quad 0{,}0625 \quad 0{,}0625 \quad 0{,}0625 \quad 0{,}0625 \quad 0{,}0625 \quad 0{,}25 \quad 0{,}0625 \quad 0{,}0625 \quad 0{,}0625]^T$$

Selanjutnya proses *boosting* dilanjutkan ke iterasi kedua. Dengan cara yang sama seperti pada iterasi pertama, dilakukan modifikasi bobot dan data *training* menggunakan algoritma SMOTE. Kemudian dilakukan *training* SVM pada data *training* baru. Sehingga diperoleh fungsi klasifikasi pada iterasi kedua

$f_2(\mathbf{x}_i) = sign\left(\mathbf{x}_i^T \begin{bmatrix} 1{,}570938 \\ -0{,}678933 \end{bmatrix} - 0{,}4512\right)$ dan garis pemisah pada proses training iterasi kedua

$$h_2(\mathbf{x}_i) = \mathbf{x}_i^T \begin{bmatrix} 1{,}570938 \\ -0{,}678933 \end{bmatrix} - 0{,}4512 = 0 \rightarrow x_2 = 2{,}3138x_1 - 0{,}6646$$



Lampiran 14. Ilustrasi Proses Klasifikasi Menggunakan SMOTEBoost-SVM (Lanjutan)

Pada proses *training* ini terdapat salah klasifikasi pada pengamatan ke-4 dan 8. *Error training* dan bobot *classifier* pada iterasi kedua masing-masing yakni $\mathbf{e}_2 = 0{,}0625 + 0{,}0625 = 0{,}125$ dan $a_2 = 0{,}9729$.

Pada akhir iterasi kemudian fungsi klasifikasi dan bobot *classifier* yang diperoleh pada tiap-tiap iterasi akan digabungkan menjadi suatu final *classifier*. Sebekumnya, bobot classifier dinormalisasi terlebih dahulu sehingga diperoleh

$$\mathbf{a} = [a_1 \quad a_2]^T = [0{,}416 \quad 0{,}584]^T.$$

Final classifier yang diperoleh yakni

$$H(\mathbf{x}_i) = sign\left(0{,}416\left(\mathbf{x}_i^T\begin{bmatrix}1{,}609283\\0{,}814254\end{bmatrix} - 0{,}5972\right) + 0{,}584\left(\mathbf{x}_i^T\begin{bmatrix}1{,}570938\\-0{,}678933\end{bmatrix} - 0{,}4512\right)\right)$$

Selanjutnya *final classifier* tersebut akan digunakan untuk mengklasifikasikan data *testing*

$$H(\mathbf{x}_1) = sign\begin{pmatrix}0{,}416\left([0{,}2228 \quad 0{,}5733]^T\begin{bmatrix}1{,}609283\\0{,}814254\end{bmatrix} - 0{,}5972\right) + \\ 0{,}584\left([0{,}2228 \quad 0{,}5733]^T\begin{bmatrix}1{,}570983\\-0{,}678933\end{bmatrix} - 0{,}4512\right)\end{pmatrix} = sign(-0.1915) = -1$$

$$H(\mathbf{x}_1) = sign\begin{pmatrix}0{,}416\left([0{,}4858 \quad -2{,}0501]^T\begin{bmatrix}1{,}609283\\0{,}814254\end{bmatrix} - 0{,}5972\right) + \\ 0{,}584\left([0{,}4858 \quad -2{,}0501]^T\begin{bmatrix}1{,}570983\\-0{,}678933\end{bmatrix} - 0{,}4512\right)\end{pmatrix} = sign(0.3774) = 1$$

Berdasarkan hasil tersebut, dapat diketahui bahwa tidak terjadi kelasahan klasifikasi *data testing* yang dilakukan oleh SMOTEBoost-SVM dengan 2 iterasi.

BIODATA PENULIS

RISKY FRASETIO WAHYU PRATAMA, lahir pada tanggal 2 Februari 1992 di Tenggarong Seberang Kab. Kutai Kartanegara, Kalimantan Timur. Penulis memulai pendidikan di Taman Kanak-kanak Sinar Bakti Tenggarong Seberang pada tahun 1997, kemudian pada tahun berikutnya melanjutkan pendidikan di Sekolah Dasar Negeri 010 Tenggarong Seberang dan tamat pada tahun 2004. Setelah lulus dari sekolah dasar, penulis melanjutkan pendidikan di Sekolah Lanjutan Tingkat Pelajar 1 Tenggarong Seberang. Pada tahun 2007 penulis melanjutkan pendidikan ke Sekolah Menengah Kejuruan 6 Samarinda dan lulus pada tahun 2010.

Pendidikan Perguruan Tinggi dimulai pada tahun 2010 dengan jalur Seleksi Nasional Masuk Perguruan Tinggi Negeri, pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman, Program Studi Statistika dan menyelesaikan program S-1 pada tahun 2015. Penulis kemudian melanjutkan pendidikan ke jenjang S-2 di Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2016 dengan mengambil jurusan Statistika di Fakultas Matematika, Komputasi, dan Sains Data (FMKSD) dan akhirnya menyelesaikan studi pada tahun 2018. Saat ini, penulis tertarik pada bidang keilmuan Machine Learning dan Data Mining sehingga mengambil Tesis pada bidang tersebut. Kritik dan Saran mengenai tulisan ini dapat dikirim ke alamat email: riskyfrasetio92@gmail.com.